**DOM ：文档对象模型**

**DOM的简介**

1. DOM是什么？
2. DOM是针对XML、HTML提供的应用编程接口
3. 我们可以将DOM看成是一种规范，当我们使用某种语言，按照这种规范，去提供具体的实现，就叫：以DOM的方式去处理问题
4. DOM能够实现语言的无关性，并且不同的开发语言对于DOM规范的实现都是不同的，也就是说JS可以用DOM的方式去处理HTML、XML，C#、PHP也可以DOM的方式去处理HTML、XML，并且他们实现DOM的方式都是不同的，当前笔记讲述的是JS以DOM的方式去处理HTML语言，也称之为：JS DOM
5. DOM方式处理标记语言的时候，是将所有的标签都加载至内存当中，并且以树状的结构去处理

-> 缺点：当节点/元素很多的时候加载就会很慢，并且对内存的负担较大

1. JS以DOM的方式去处理HTML的时候，会把每一个标签都当成一个对象去处理，这些对象我们称之为DOM对象

**与JS对象区分开来**

1. DOM对象拥有着：事件、属性、函数

-> 因为标签具有与用户交互的功能，所以拥有事件

1. JS自定义对象拥有着：属性、函数

-> 因为JS自定义对象只是在内存中运行的代码，所以没有事件

1. 使用JS进行DOM操作，主要是包含着两个方面
2. Browser对象：**BOM**

-> 在一个浏览器中的外边框，如收藏夹、URL地址…，都是属于BOM

-> 操作的是窗口：window.\*\*\*\*

1. HTML DOM对象：**DOM（狭义的）**

-> 一个网站的页面，即<html></html>标签中的所有内容，都是属于HTML DOM对象

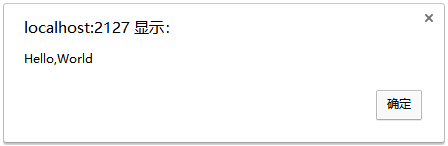
-> 操作的是文档：window.document.\*\*\*\*

**BOM详解**

1. **window对象的三种弹出浏览器对话框的函数**
2. alert ( "任意字符串" )

-> 弹出网页提示框

-> 代码规范与实现效果

1. confirm ( "任意字符串" )

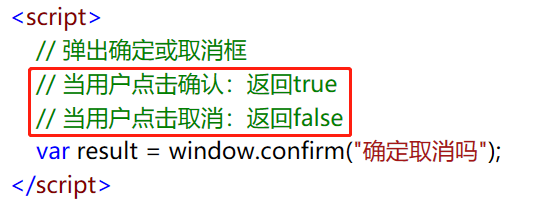
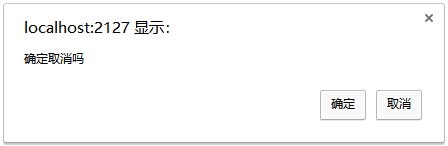
-> 弹出确定或取消框

-> 返回值：

-> 点击确认：返回true

-> 点击取消：返回fasle

-> 代码规范与实现效果

1. prompt ( "提示文本字符串" , "文本框默认值字符串" )

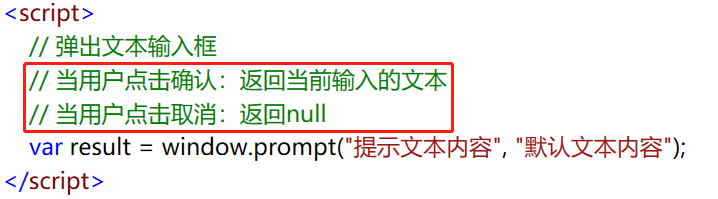
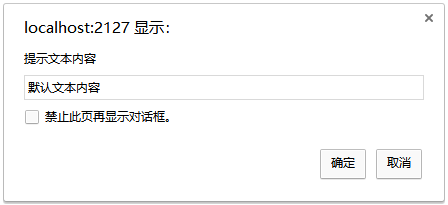
-> 弹出文本输入框

-> 返回值：

-> 点击确定：返回当前输入的文本

-> 点击取消：返回null

-> 代码规范与实现效果

1. **location对象**

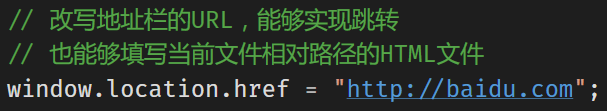
-> location：表示记录当前浏览器窗口URL信息的一个对象

-> location对象是window对象的一种属性

1. href属性

-> 获取或设置当前网页的URL地址

-> 代码规范与实现效果

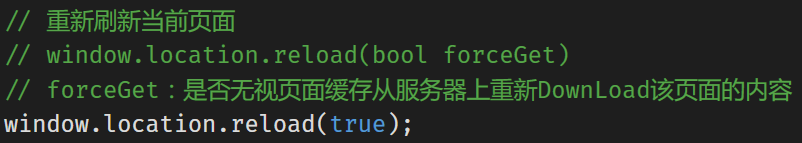
 QQ截图20180430163415

1. reload ( bool reload )

-> 重新获取当前页面的内容

-> bool reload：如果把该方法的参数设置为 true，那么无论文档的最后修改日期是什么，它都会绕过缓存，从服务器上重新下载该文档

-> 代码规范与实现



1. **定时器与延时器**
2. Intervalled setInterval ( object handler , object timeout )

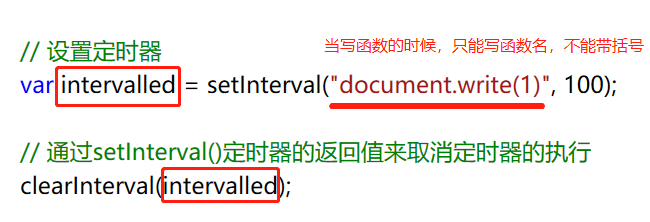
-> 定时器

-> obejct handler：表示定时器需要执行的代码，如果直接写代码，则需要在代码里加入 "" ，如果直接写函数，则不能带函数的括号，只能写函数名

-> object timeout：表示定时器间隔多久执行一次 handler 代码，单位为毫秒

-> Intervalled返回值：我们可以通过指定clearInterval ( ) 函数的参数为定时器的返回值Intervalled，以达到关闭我们指定的定时器的目的

-> 代码规范与实现



1. Intervalled setTimeout( object handler , object timeout )

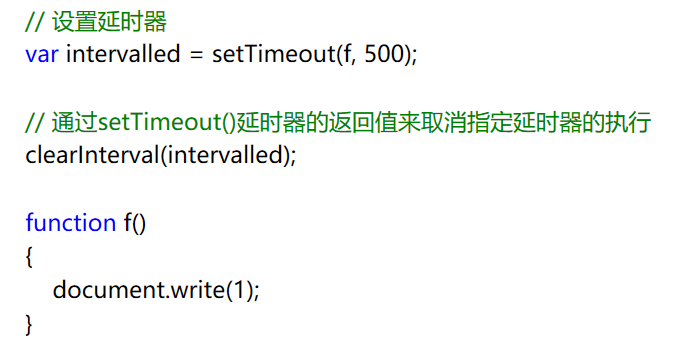
-> 延时器

-> obejct handler：表示延时器需要执行的代码，如果直接写代码，则需要在代码里加入 "" ，如果直接写函数，则不能带函数的括号，只能写函数名

-> object timeout：表示延时器间隔多久执行一次 handler 代码，单位为毫秒

-> Intervalled返回值：我们可以通过指定clearInterval ( ) 函数的参数为定时器的返回值Intervalled，以达到关闭我们指定的延时器的目的

-> 代码规范与实现

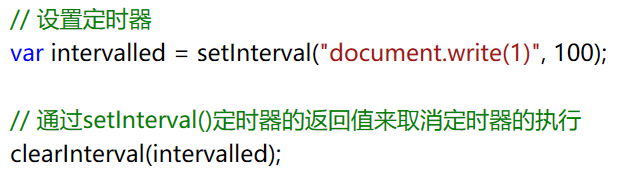


1. clearInterval ( object Intervalled )

-> 清空定时器/延时器

-> object Intervalled：定时器/延时器函数的返回值Intervalled，通过定时器/延时器的返回值来取消指定的定时器/延时器的执行

-> 代码规范与实现



1. **window对象的一些属性（具体查阅API文档）**



**HTML DOM详解**

1. **document对象**

**-> document：当前页面中所有HTML标签所构成的一个对象**

**-> document对象是window对象的一种属性**

1. getELementById ( string ID ) /GetElementsByClassName ( string Class ) /…

-> 获取指定的HTML标签对象

-> 当我们获取了页面中的HTML标签对象，我们就可以通过HTML标签对象的属性来实现实现：在JS中访问HTML标签的属性或为HTML标签注册DOM事件等等

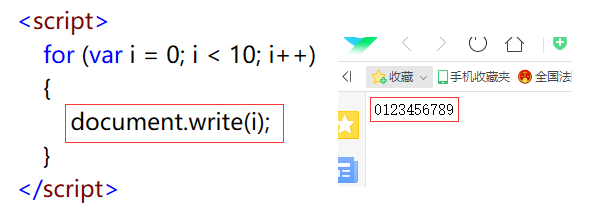
-> 代码规范与实现



1. write ( )

-> 通过document对象向浏览器文档部分写入指定内容

-> 代码规范与实现



**HTML DOM动态操作节点的方式**

1. **createElement ( string tagName )**

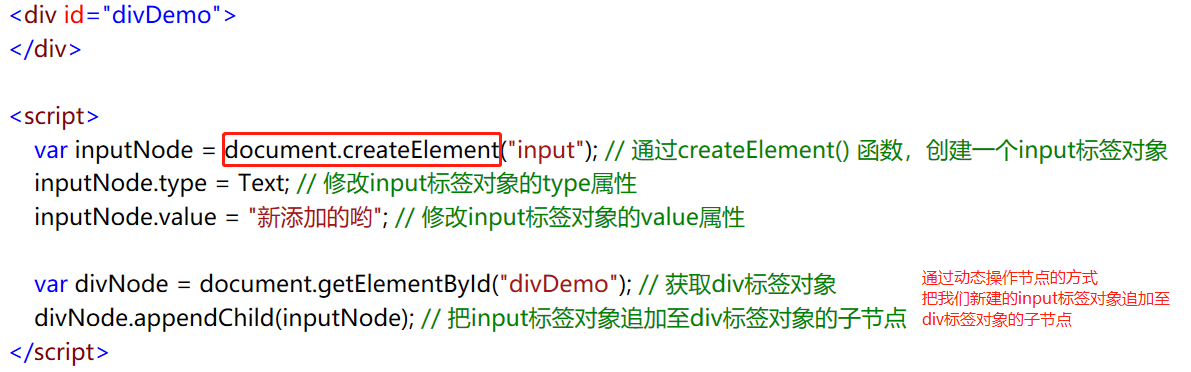
-> 创建HTML标签对象

-> tagName：HTML标签名

-> 我们可以通过该函数去创建指定HTML标签名的HTML标签对象，例如，我指定了参数tagName为 "input" ，那么就会返回一个 input标签 的对象

-> 当我们去创建并返回了一个HTML标签对象，我们就能修改该HTML标签对象的属性，并通过动态操作节点的方式把该标签对象添加至HTML标签当中

-> 代码规范与实现



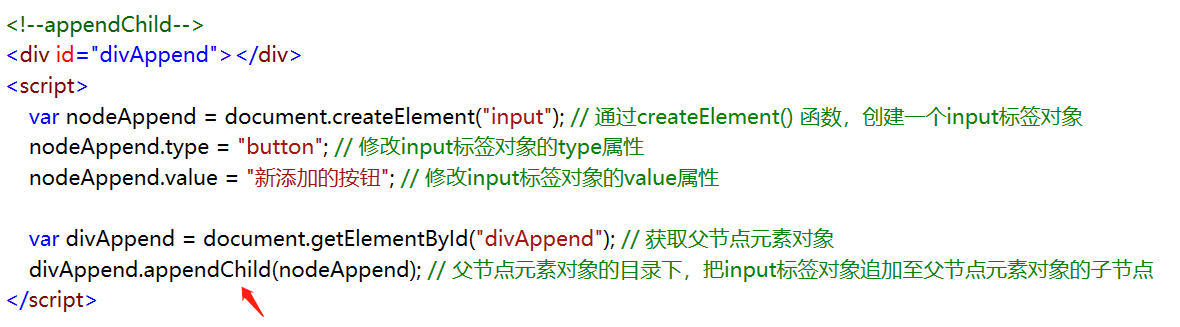
1. **HTML标签对象.appendChild ( Node node ) 函数**

-> 在当前父节点HTML标签对象的目录下，添加元素对象node为父节点的子节点

-> HTML标签对象：父节点对象

-> Node node：需要添加的子节点

-> 代码规范与实现



1. **HTML标签对象.insertbefore( Node newNode , Node oldNode ) 函数**

-> 在当前父节点HTML标签对象的目录下，插入新元素对象newNode至旧元素对象oldNode的前面

-> HTML标签对象：父节点对象

-> newNode：新元素对象

-> oldNode：旧元素对象

-> 代码规范与实现



1. **Node[] nodes HTML标签对象.childNodes 属性**

-> 获取当前父节点目录下的所有子节点

-> HTML标签对象：父节点对象

-> ChildNodes：父节点对象的读写属性，用于获取当前父节目录下的所有子节点

-> nodes：返回的子节点元素集合

-> **其实每个子节点都有着一个与之对应的text节点，该节点用于描述子节点的一些文本信息**

-> 代码规范与实现



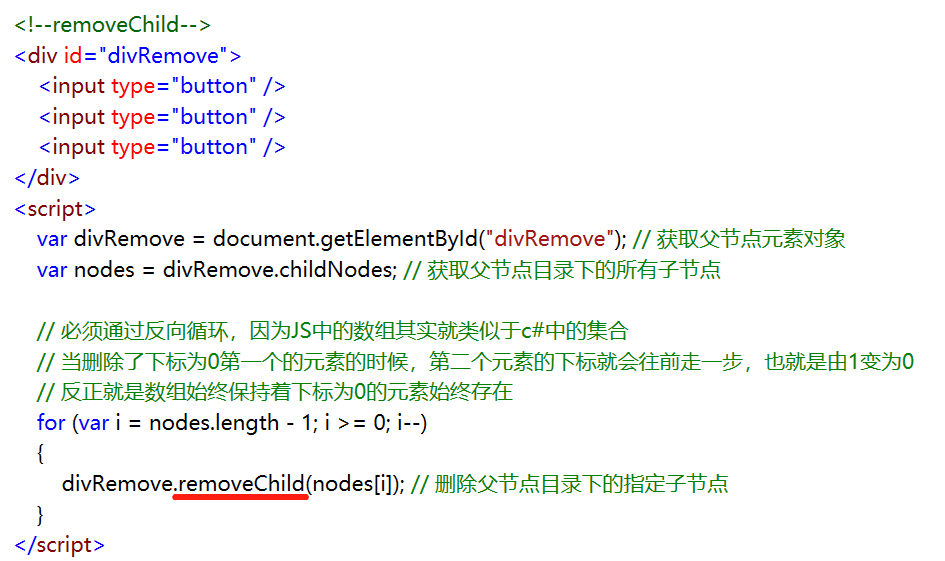
1. **HTML标签对象.removeChild( Node node ) 函数**

-> 删除当前父节点目录下的指定子节点

-> HTML标签对象：父节点对象

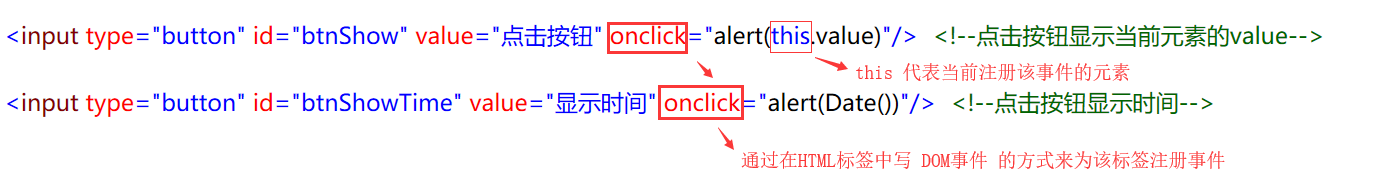
-> node：需要删除的子节点

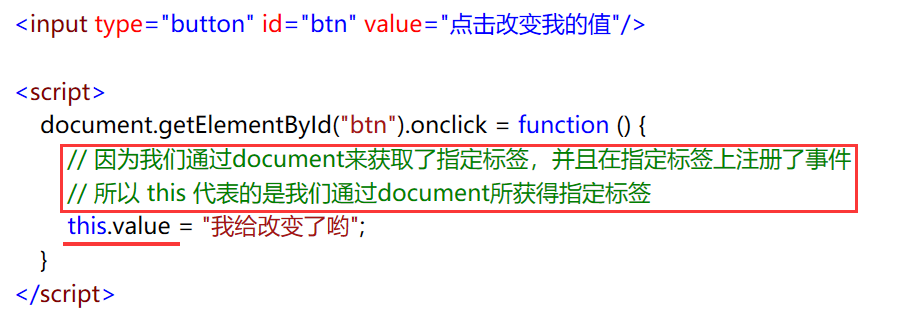
-> 代码规范与实现



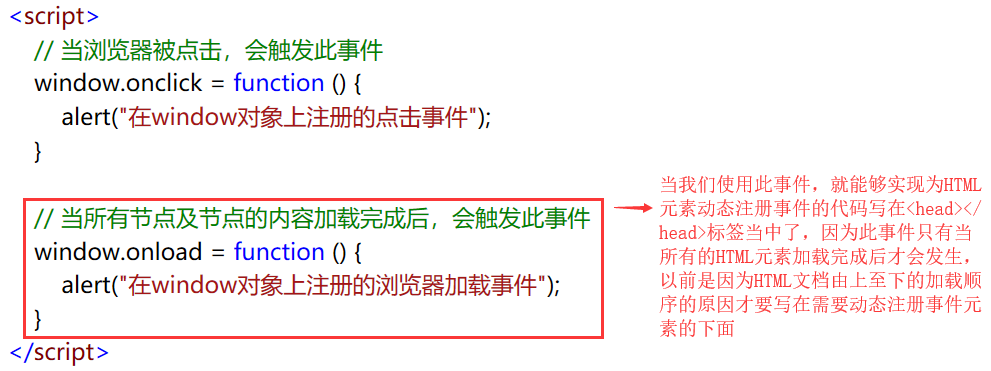
**HTML DOM事件**

1. **DOM事件的注册：**
2. **注册HTML标签的DOM事件，两种方式**
3. 直接在HTML元素上注册事件（不推荐）

 2） 在JS脚本中，通过HTML DOM获得指定标签，并注册事件（推荐：实现了JS与HTML的分离）

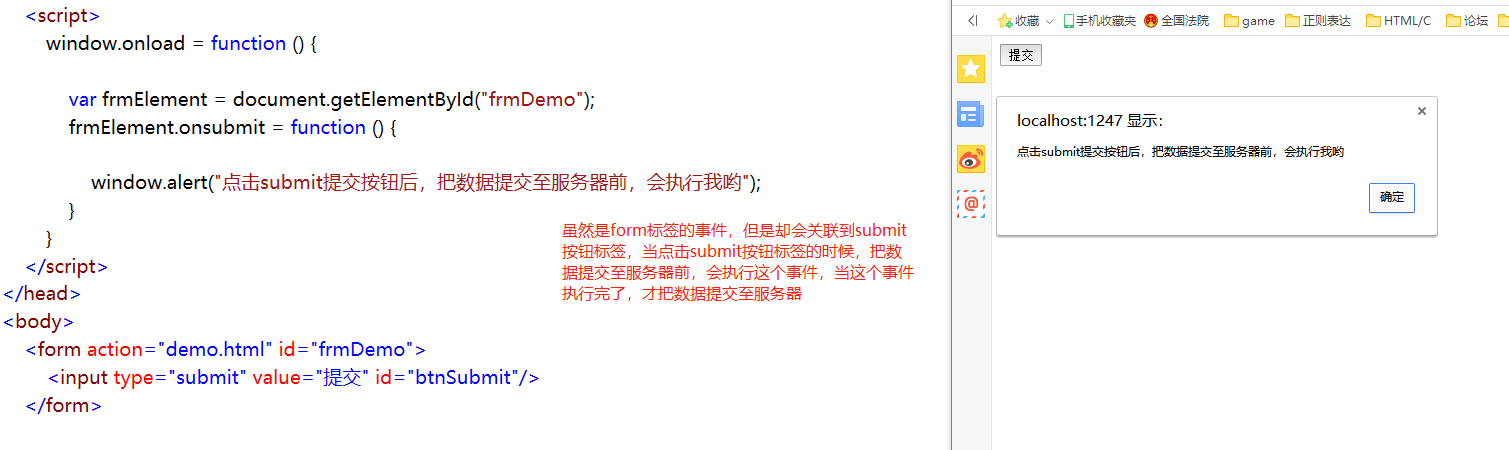


1. **window对象注册DOM事件**



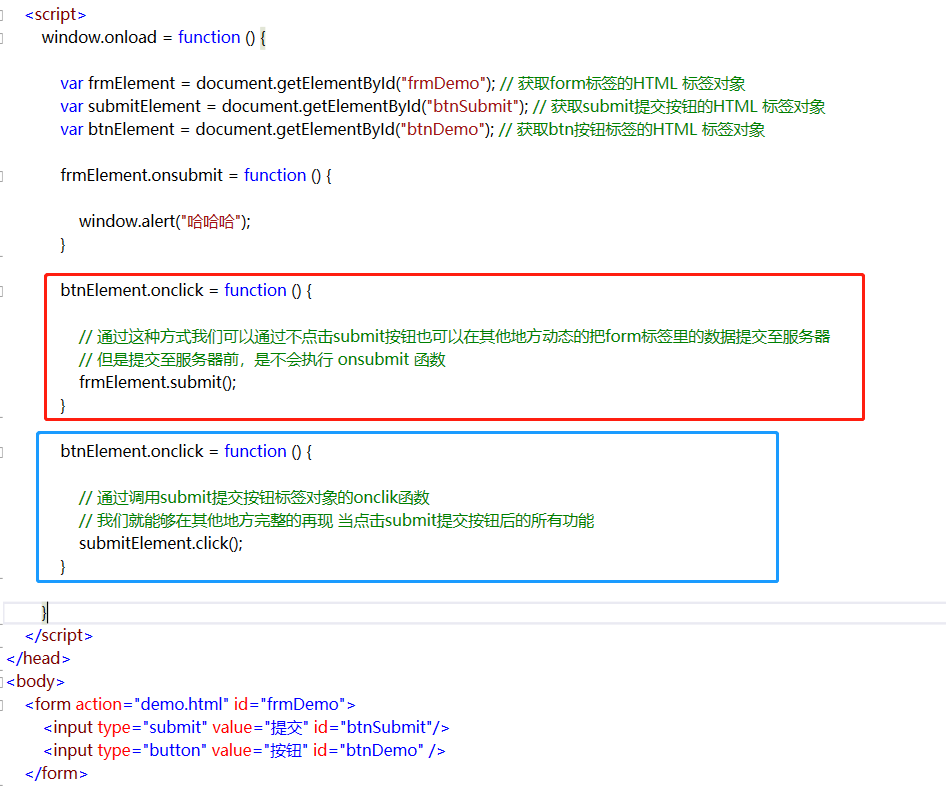
1. **form标签对象的onsubmit事件**

-> 当点击<input type= "submit" />submit提交按钮后，<form></form>标签内部的数据便会提交至服务器，当指定了该事件所执行的函数，那么在**数据提交至至服务器前**，会执行onsubmit事件

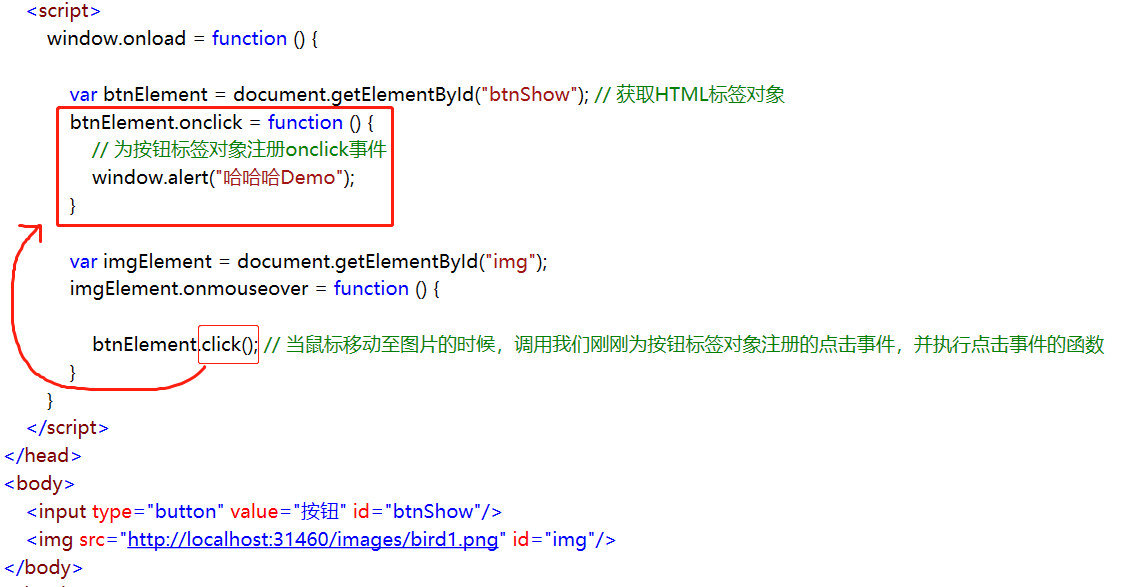
 -> 当我们在onsubmit事件所指向的函数中：return false，则不会把数据提交至服务器，也就是不会进行页面的跳转



-> 我们可以通过在其他地方调用form标签对象的submit()函数，来在其他地方动态的实现把form标签里的数据提交至服务器地址，但是，**通过这种方式是不会在form标签里的数据提交至服务器前执行onsubmit事件**，**除非不调用form标签对象的submit函数而是通过调用submit提交按钮标签对象的click函数模拟提交按钮的点击事件，才能在其他地方完整的的实现点击submit提交按钮后的所有功能（即把form标签内部的数据提交至服务器地址，又触发了onsubmit事件）**



1. **当我们已经为一个按钮注册过了onclick事件，那么我们可以通过该按钮的HTML Element的click()函数在其他地方调用（模拟）该按钮的点击事件**

****

1. **DOM中支持的事件（参考W3CSchool手册）**

****

1. **DOM事件中的 event/e 参数：**

**-> 该参数类似于C#控件的事件所执行函数的参数中的 eventArgs e**

**-> 该参数主要用于，当事件执行的时候，能够通过该参数的属性值获取该事件执行者的一些额外的、附加的信息**

**-> 事件执行者的不同，通过参数 event/e 的属性获取的事件执行者的额外的信息也各不相同**

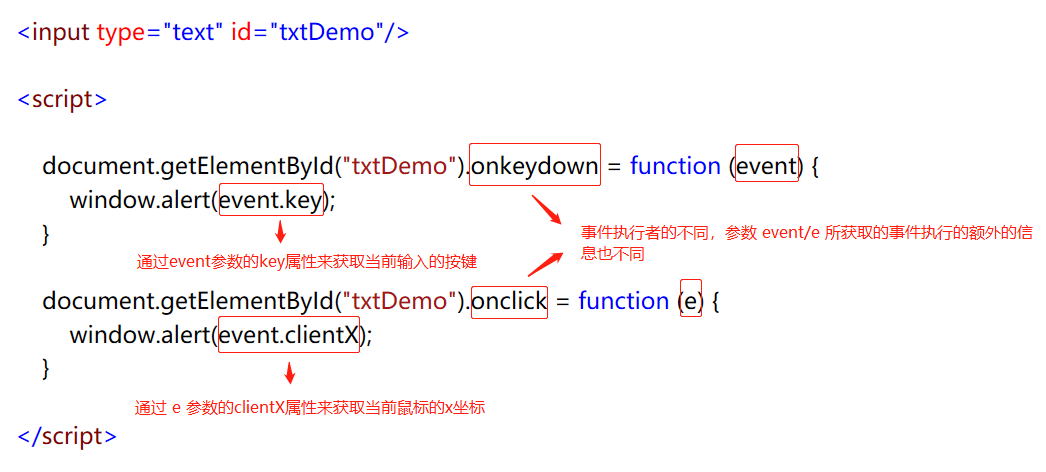
1. **Event参数的使用**

-> 我们直接在DOM事件所执行的函数的参数中写 event/e（推荐）,然后在函数的方法体中我们就能通过参数 Event/e 的属性来获取该事件执行者的一些额外的信息

-> 火狐浏览器不支持event参数，但是支持e参数

**-> 由于浏览器内部实现DOM的不同，不同浏览器支持event都有差异，所以建议参数写成e的形式**

-> 代码规范与实现



1. **event参数中的属性具体参考W3CSchool**

****

**HTML标签对象的一些属性、函数 ( 通过HTML DOM获取到HTML标签对象 )**

1. **HTML标签对象.InnerText/textContent 属性**

-> 通过HTML标签对象的两个属性，获取HTML标签内部的文本内容

-> 火狐浏览器只支持textContent属性的实现

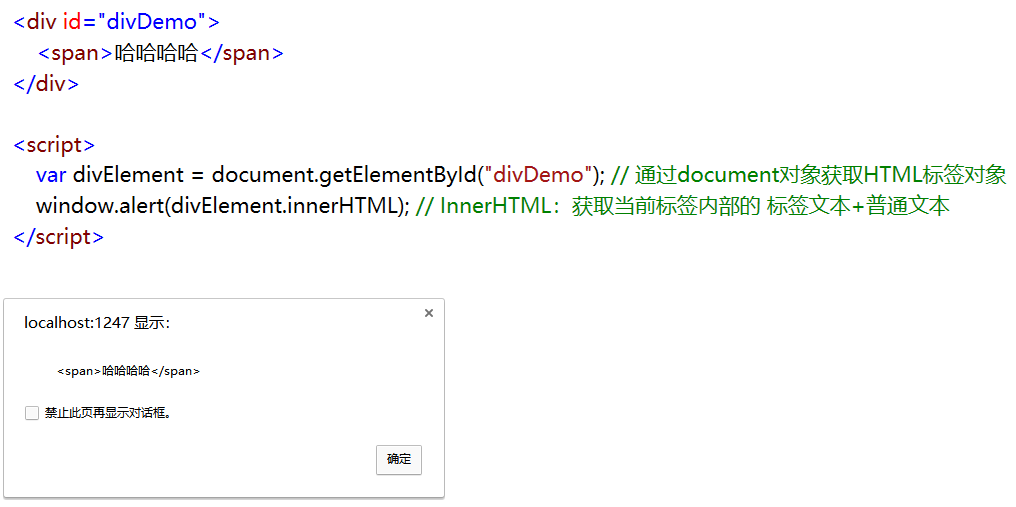
-> 代码规范与实现



1. **HTML标签对象.InnerHTML 属性**

-> 通过HTML标签对象的InnerHTML属性，获取HTML标签内部的HTML标签文本+普通文本

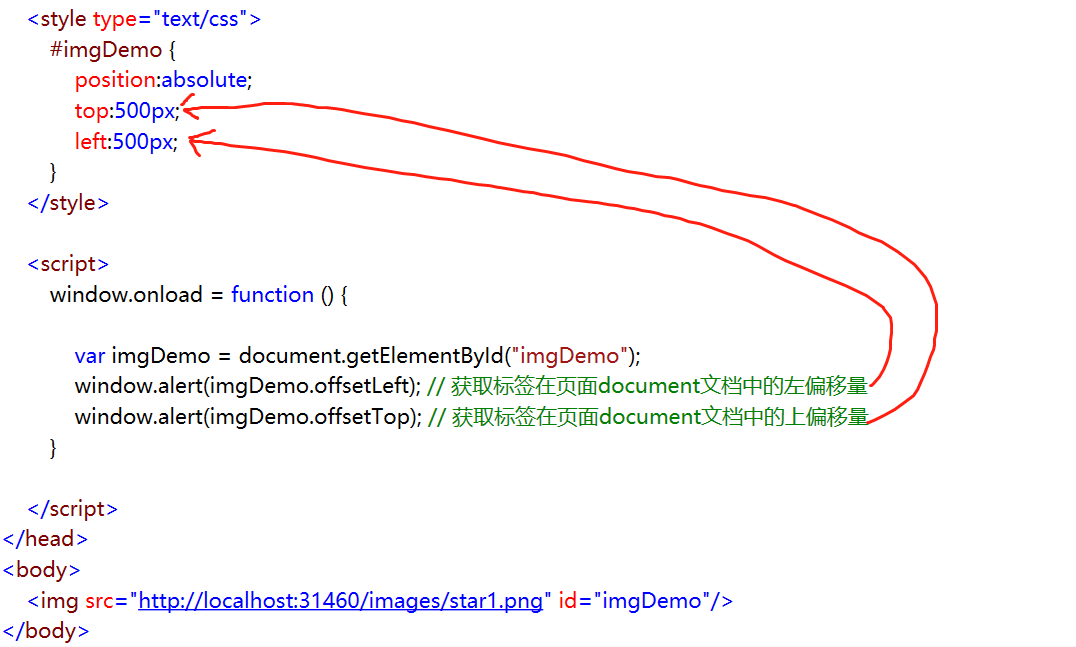
-> 代码规范与实现



1. **HTML标签对象.offsetLeft / offsetTop 属性**

-> 通过HTML标签对象的offsetLeft / offsetTop 属性，获取HTML标签在页面的document文档中的 左偏移量 / 上偏移量

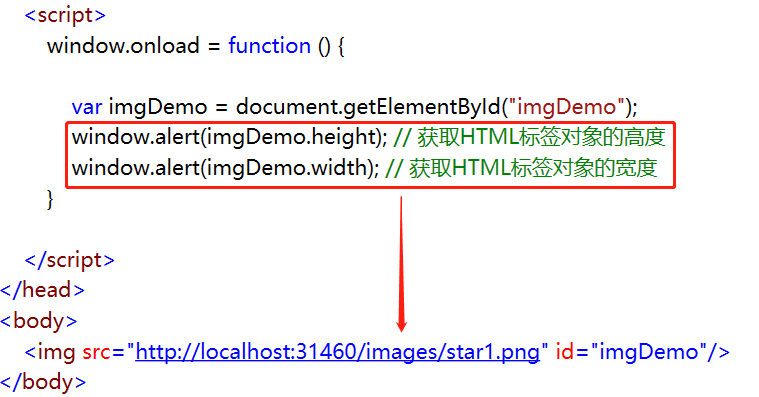
-> 代码规范与实现



1. **HTML标签对象.height / width属性**

-> 通过HTML标签对象height / width的属性，获取HTML标签对象的 高度 / 宽度

-> 代码规范与实现

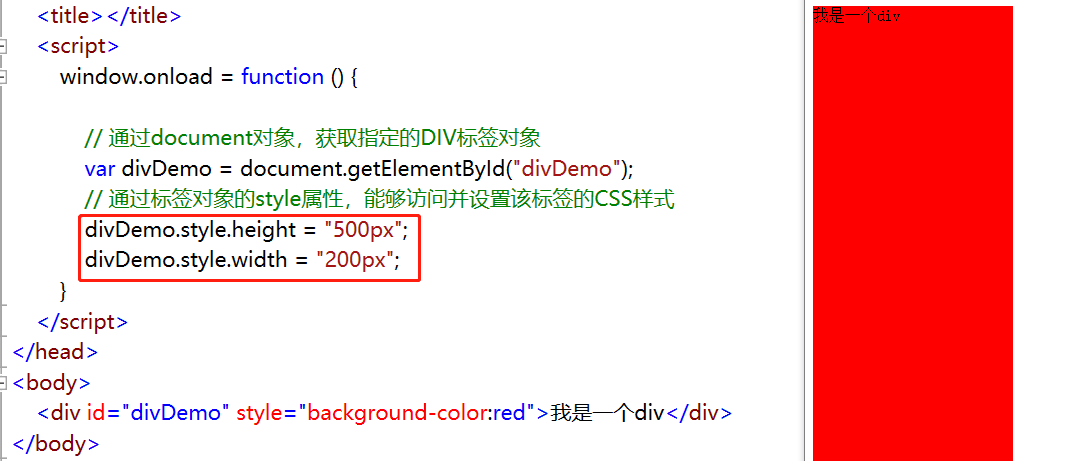


1. **HTML标签对象.style.CSS样式的属性**

-> 通过HTML标签对象的style属性，能够在JS DOM中访问并设置指定标签的CSS样式

**-> 代码规范与实现**

1. 普通CSS样式属性的代码规范与实现



1. 特例CSS样式属性的代码规范与实现
2. 带有 - 号的CSS样式属性：我们通过 JS DOM 像操作该样式属性的时候是访问不到的，因为JS的属性命名规范不能带有 - 号，所以，需要去掉 - 号，并且 - 号后面的单词首字母大写，通过这种书写方式，就能在 JS DOM中访问到特例的CSS样式属性



1. float属性：在JS DOM虽然中可以直接通过该CSS样式属性名直接访问和修改该标签的float属性，但是火狐浏览器不支持float的书写方式去修改该标签的float样式，火狐浏览器及其他浏览器支持的是 cssFloat 这种属性名的书写方式



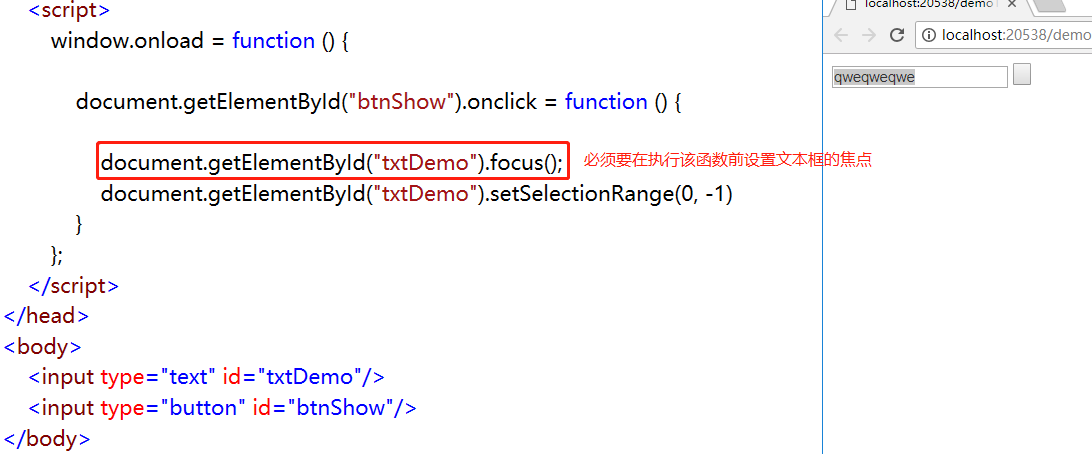
1. **HTML标签对象.setSelectionRanage ( startIndex , endIndex )**

-> 设置text或textarea的 statrIndex至endIndex位置中的文本为选中高亮状态

-> **必须要在该函数执行前，获取到text或textarea的焦点才能有效果**

-> endIndex的位置不包含在设置文本高亮状态的位置，故需要+1/-1

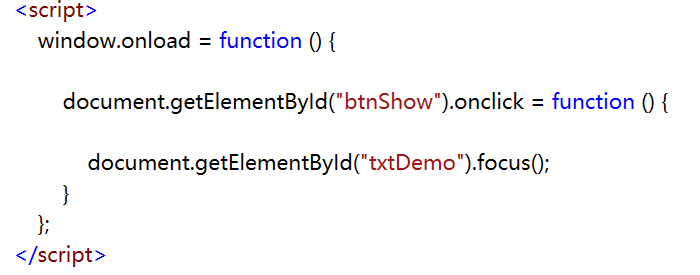
-> 代码规范与实现



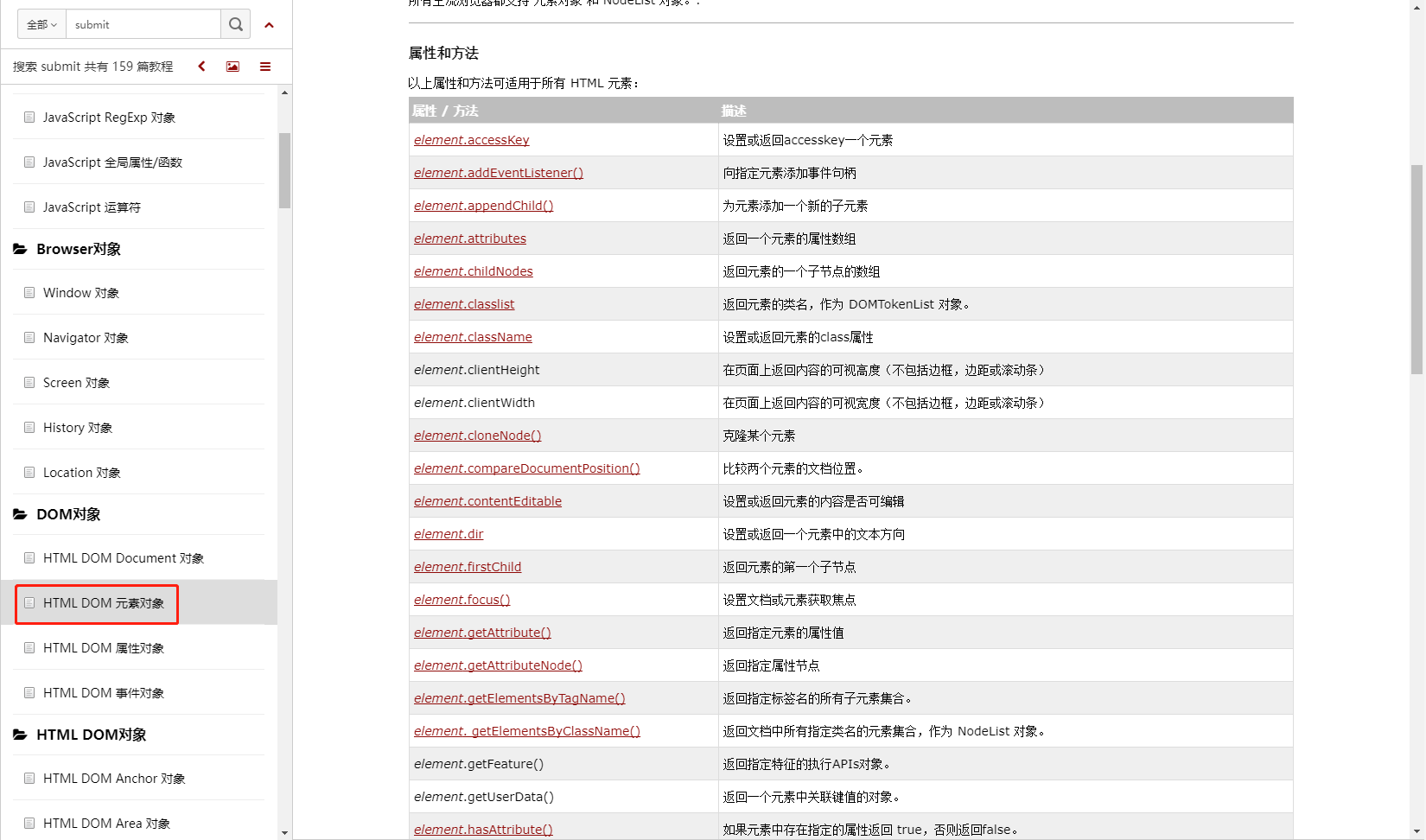
1. **HTML标签对象.focus()**

-> 获取指定HTML标签的焦点

-> 代码规范与实现



1. **HTML标签对象支持的属性和函数（参考W3CSchool）**

****