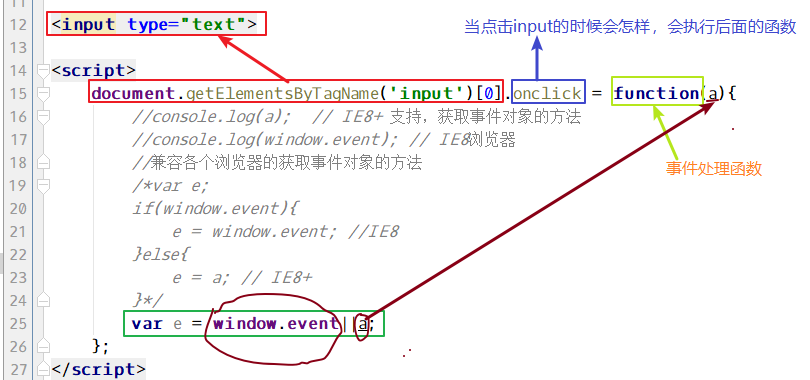
# 一、昨日反馈

1、事件对象

IE浏览器获取事件对象： window.event;

非IE：使用的是事件处理函数的形参

兼容性的写法：var e = window.event||evt;



2、获取页面中的节点

document.getEle….By…

document.querySelector('css选择器');

document.querySelectorAll('css选择器');



# 二、今日目标

1、能够对属性类型的节点进行增删改查操作(重点)

2、能够学会使用innerHTML和innerText（重点）

3、知道通过nodeValue可以获得文本类型的节点的值

4、知道nodeType表示什么

5、能够掌握三种绑定事件的方式（重点）

6、能够知道事件处理函数中this表示什么（重点、难点）

7、能够知道如何移除事件

8、能够通过事件对象的keyCode获取键盘的对应值

9、了解什么是冒泡事件及会阻止冒泡事件的发生

10、能够会用一种办法阻止标签的默认行为

# 三、DOM之属性节点操作

属性不能单独存在，一定属于一个标签。所有，操作属性，前提必须找到标签。

3.1、获取元素的属性

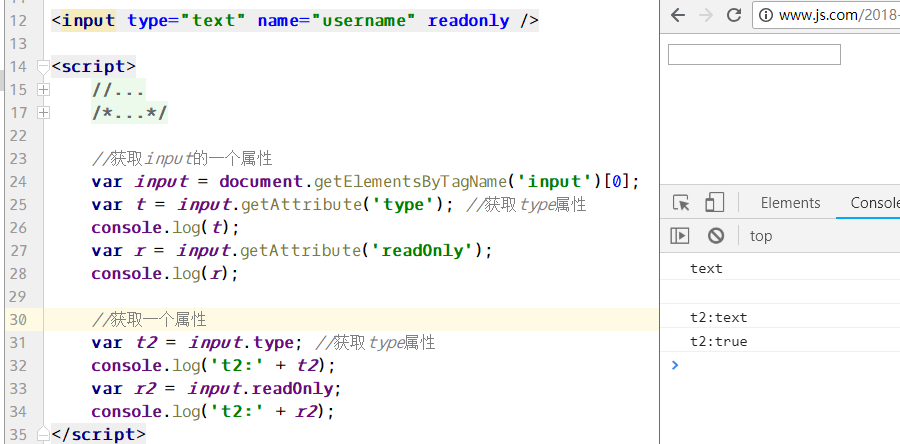
|  |  |  |  |
| --- | --- | --- | --- |
| 语法 | 描述 | 特点 | 支持 |
| 元素节点.attributes | 获取元素的所有属性 | - | 所有浏览器 |
| 元素节点.getAttribute(属性名) | 获取指定属性的值 | 标准的获取已经存在的属性值的方法 | 所有浏览器 |
| 元素节点.属性名 | 获取指定属性的值 | 获取不存在的属性值时比较好用 | 所有浏览器 |

attributes:



getAttribute(); ：获取元素的属性。只能获取有具体值的属性

元素.属性：获取元素的属性，能够获取类似于readOnly、checked等属性



3.2、添加/修改元素的属性

|  |  |  |  |
| --- | --- | --- | --- |
| 语法 | 描述 | 特点 | 支持 |
| 元素节点.setAttribute(属性名，属性值) | 设置(添加/修改)元素的所有属性 | 属性不存在表示添加，属性存在表示修改 | 所有浏览器 |
| 元素节点.属性名 = 值 | 设置(添加/修改)元素的所有属性 | 属性不存在表示添加，属性存在表示修改 | 所有浏览器 |

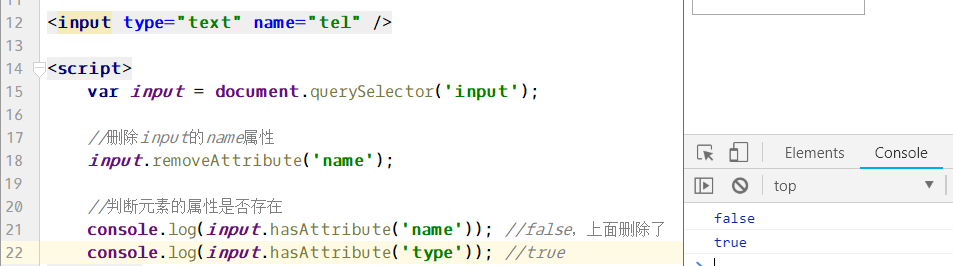




**如果要设置checked、readonly、disabled，则需要设置它的值为true或false**。

3.3、删除元素的属性

元素节点.removeAttribute(属性名) 删除一个属性



3.4、判断元素是否有哪个属性

elementNode.hasAttribute(属性名) 检测是否有某个属性，有返回true，没有返回false

检测，看3.3代码。

# 总结：

节点分为元素类型的节点和属性类型的节点，也包括文本类型的节点。

元素类型的节点：

增： document.createElement()、document.createTextNode()、父节点.appendChild()、父节点.insertBefore()

删：父节点.removeChild();

改：父节点.replaceChild();

查：document.getEle….By… document.querySelector()

属性类型的节点：

增：元素.setAttribute(属性名, 值); 元素.属性名 = 值;

删：元素.removeAttribute(属性名);

改：元素.setAttribute(属性名, 值); 元素.属性名 = 值;

查：元素.attributes; 元素.getAttribute(属性名); 元素.属性名;

# 四、DOM对象的通用属性

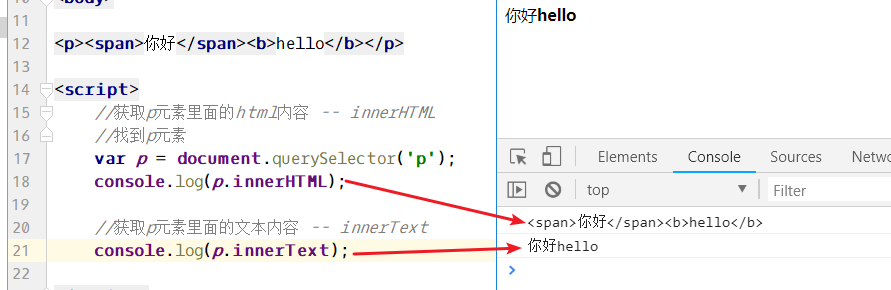
**innerHTML**

​ 获取/设置元素里的html内容

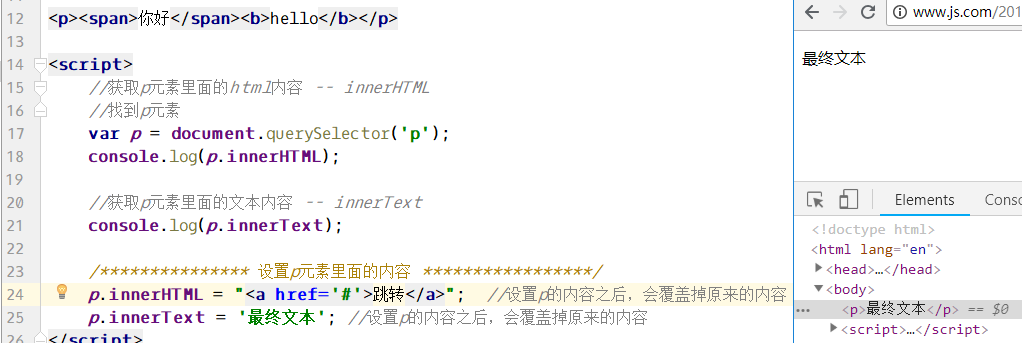
**innerText**

​ 获取/设置元素里面的文本内容

**演示获取标签里面内容的代码**：



**演示设置标签内容的代码**：



**nodeName**

​ nodeName 是只读的

​ 元素节点的 nodeName 与标签名相同

​ 属性节点的 nodeName 与属性名相同

​ 文本节点的 nodeName 始终是 #text

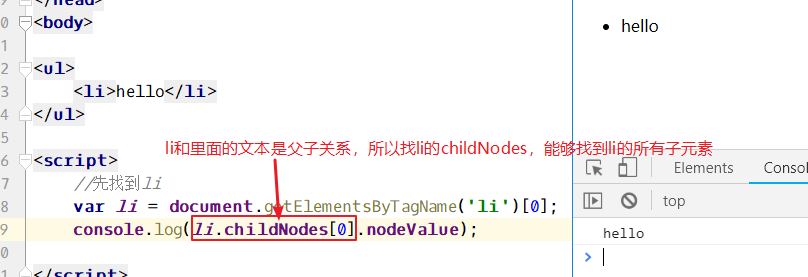
​ 文档节点的 nodeName 始终是 #document

**nodeValue**

​ 元素节点的 nodeValue 是 undefined 或 null

​ 文本节点的 nodeValue 是文本本身

​ 属性节点的 nodeValue 是属性值



**nodeType**：调用nodeType属性会得到一个数字，这个数字表示节点的类型

​ 元素 1

​ 属性 2

​ 文本 3

​ 注释 8

​ 文档 9

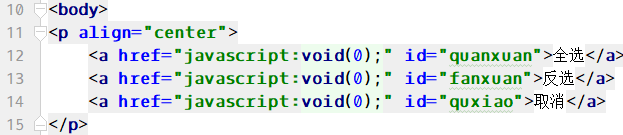
下面代码演示了使用nodeValue和nodeType的例子：

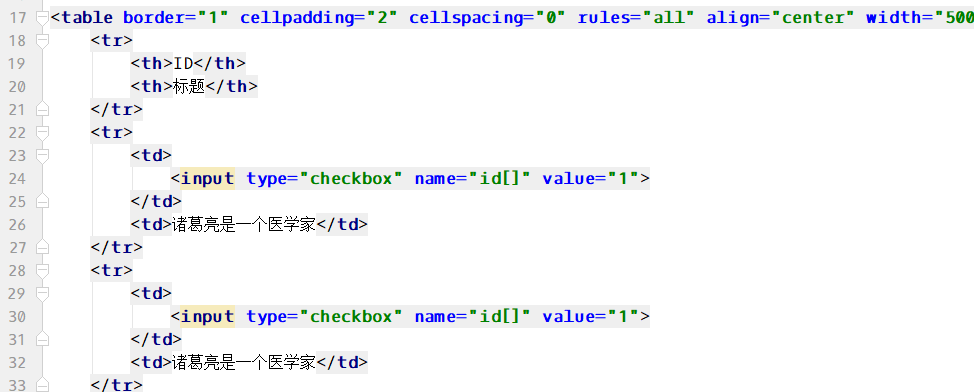
下面代码演示了nodeType的其他用法：

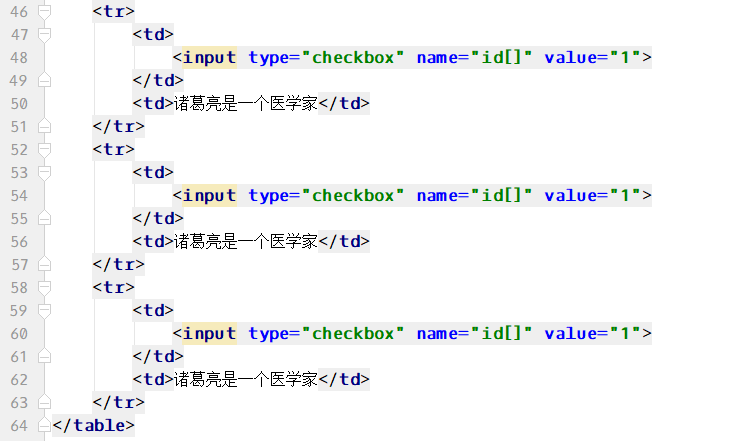


# 案例 -- 全选、反选、取消

HTML代码：

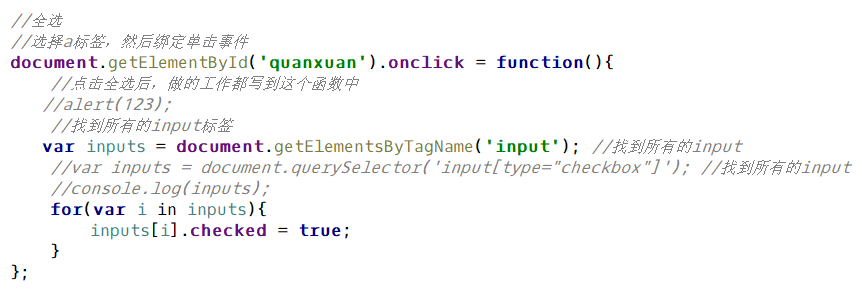




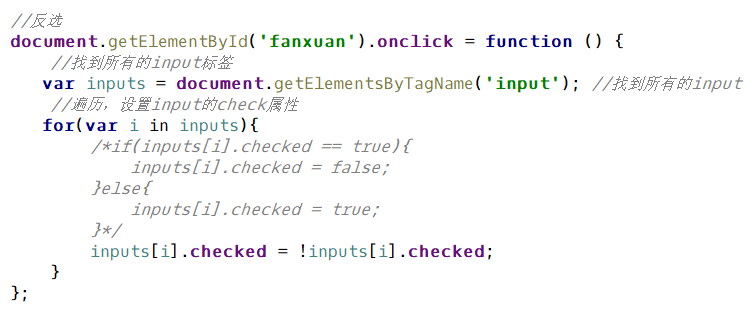


JS代码：

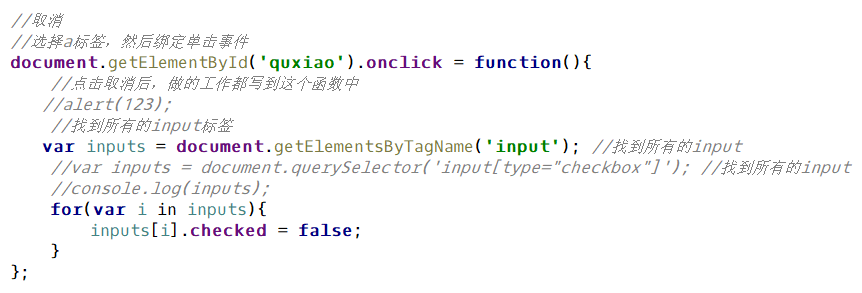
全选：



反选：



取消：



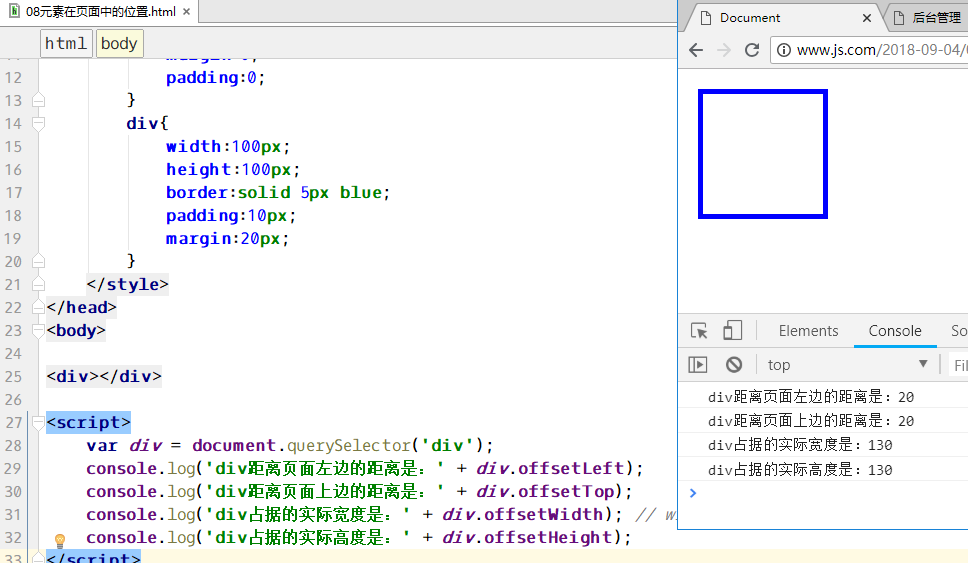
# 五、DOM获取元素的位置

offsetLeft 元素在网页中水平坐标值

offsetTop 元素在网页中垂直坐标值

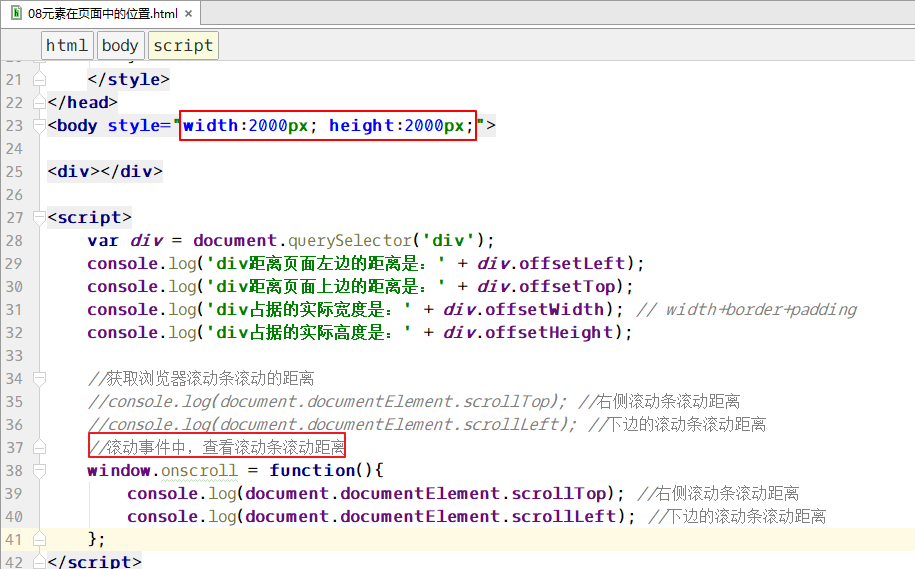
offsetWidth 元素在页面中占据的宽度

offsetHeight 元素在页面中占据的高度



scrollLeft 滚动条在容器中水平滚动的距离，多用于浏览器的滚动条

scrollTop 滚动条在容器中垂直滚动的距离，多用于浏览器的滚动条



**这部分内容一般用于窗口显示，异步加载等**。目前可以作为了解的内容，等学习了jQuery，jQuery中有封装好的兼容各个浏览器的方法。

# 六、事件绑定与移除

事件绑定就是添加事件，那么为谁添加事件呢？可以为页面中的所有标签（元素、节点）添加事件，大到整个文档（document），小到其中一个标签都可以添加事件。

在为一个节点添加事件的时候，有三种方式：

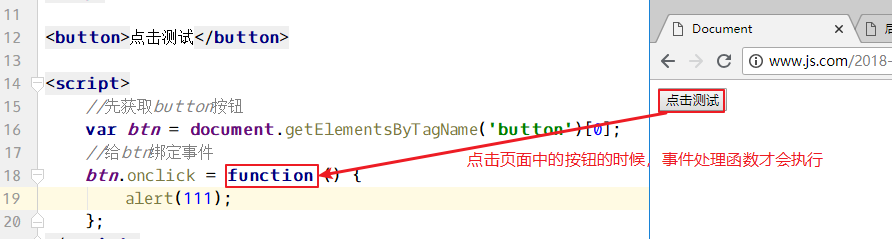
* 直接在标签中添加，即使用onclick、onmouseover等HTML属性；
* DOM的方式添加，即通过dom方法获取节点，然后为其绑定事件；
* 使用addEventListener或attachEvent（IE8及更低版本浏览器）。

**为一个标签添加事件之后，还可以删除这个事件**。

6.1、为DOM对象添加事件

最基本的语法：

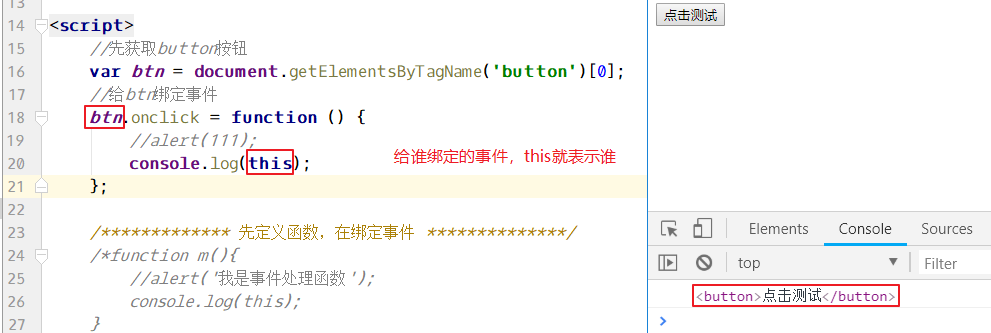
**DOM对象.事件 = 处理函数;**



有些时候，可以先定义处理函数，然后在绑定事件：



**事件处理函数中，this表示什么**？

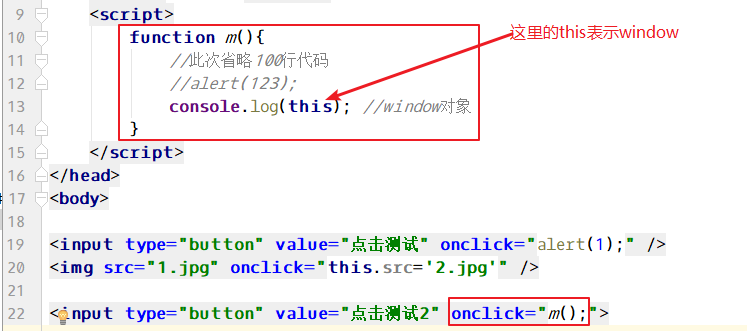


6.2、直接在HTML标签中添加事件

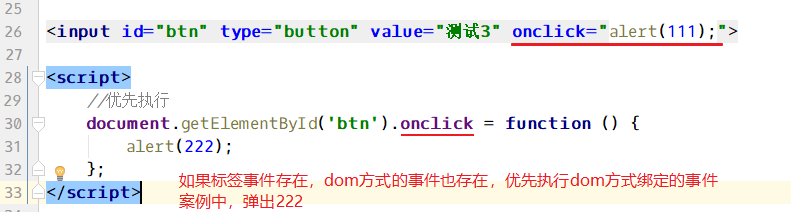
通过html标签的on….属性（onclick、onmouseover…），来添加事件。



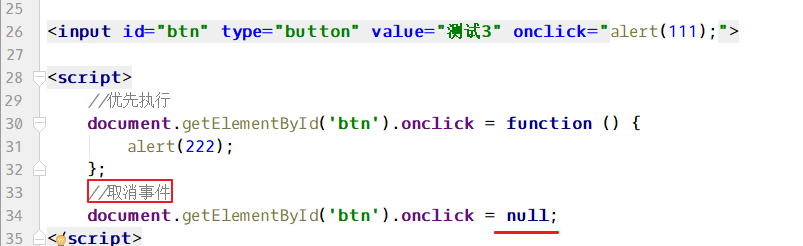
如果说，单击按钮的时候，要执行的代码非常多，则可以将代码封装成函数，然后onclick的时候，调用函数即可，函数中的this是window对象。



**如果上面两种绑定事件的方式都存在，执行哪一种**？



**上面两种方式绑定事件，都可以通过设置事件处理函数为null来去取事件**：

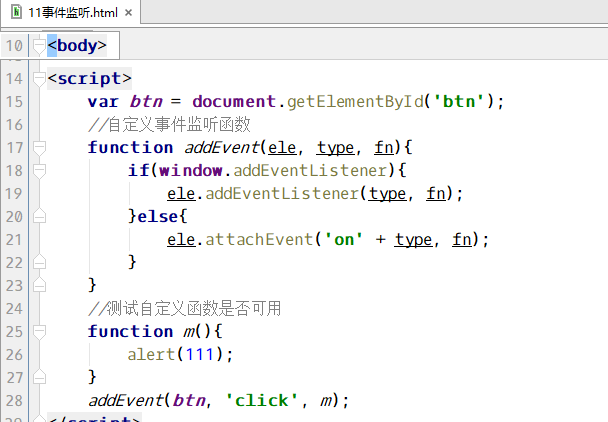


6.3、使用addEventListener或attachEvent方法添加

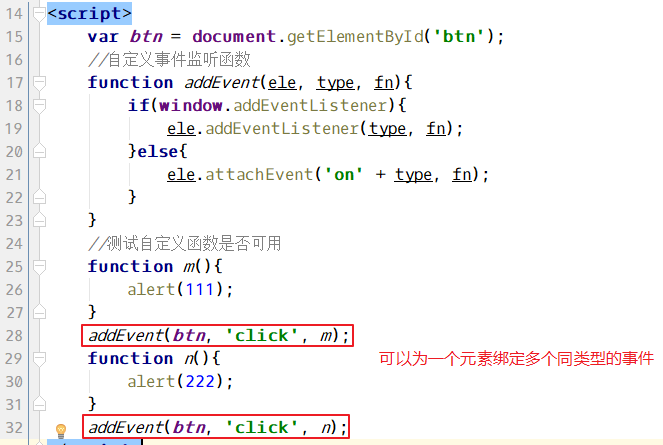
使用下面的方法为DOM对象添加事件，也叫做事件监听。

* IE浏览器中用：元素节点.attachEvent(事件名, 事件处理函数)；
* 火狐浏览器用：元素节点.addEventListener(事件名, 事件处理函数)；// 事件名不要带on

为了兼容各个浏览器，可以自定义一个兼容的函数，来为元素绑定事件：



**事件监听方式添加的事件，特点是可以为一个元素绑定多个同类型的事件**。

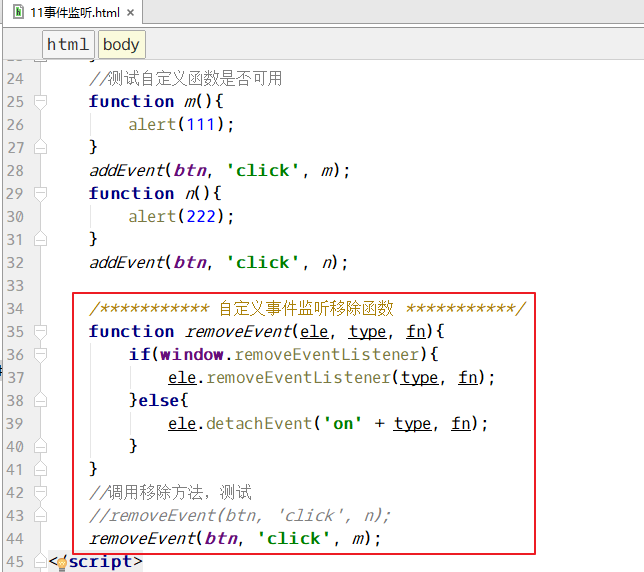


通过事件监听方法添加的事件，也可以被移除，移除的方法是：

火狐：元素节点**.**removeEventListener(事件名, 处理函数); //事件名不要带on

IE：元素节点**.**dettachEvent(事件名 处理函数);

也可以自定义一个兼容各个浏览器的移除事件监听的方法：



# 七、事件对象相关属性和方法使用

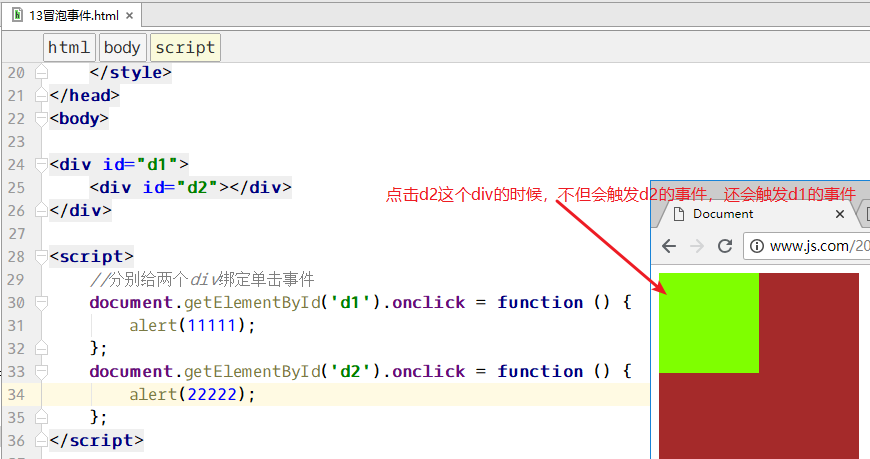
事件对象存在于事件处理函数中。不同的浏览器获取事件对象的方式也不同，IE始终使用window.event来当做事件对象，非IE浏览器使用事件处理函数的形参当做事件对象。

7.1、获取键盘的keyCode值



7.2、冒泡事件及阻止冒泡事件的发生

**什么是冒泡事件？**



上述小例子中，点击元素的子元素，会透过子元素触发元素本身的事件，这就是冒泡事件。

有些时候要的就是这个效果，可以不做任何处理；有些时候不希望有冒泡事件发生，那么可以通过事件对象的方法来阻止冒泡的发生。

阻止冒泡事件的发生有：

* 标准浏览器使用 evt.stopPropagation(); //evt指的是事件对象
* IE内核浏览器使用 window.event.cancelBubble = true;

可以封装一个兼容各个浏览器的阻止冒泡的方法：



7.3、阻止标签的默认行为

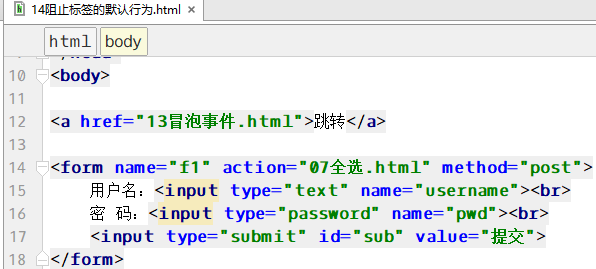
默认行为就是html标签的一些默认行为，比如点击a标签会跳转，比如点击了submit按钮表单会提交。这些都属于标签的默认行为。

有些时候，点击了a标签或者submit按钮后不希望执行标签的默认行为，这时候就需要阻止默认行为。

阻止默认行为：

标准浏览器：evt.preventDefault();

IE内核浏览器：window.event.returnValue = false;

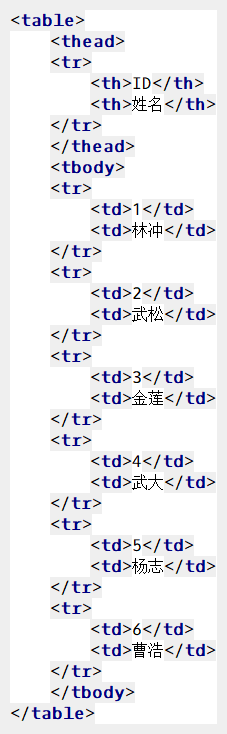


写兼容各个浏览器的阻止标签默认行为的方法：



# 八、案例--可编辑的表格

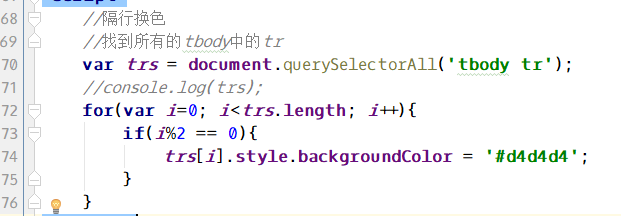
1、完成html+css



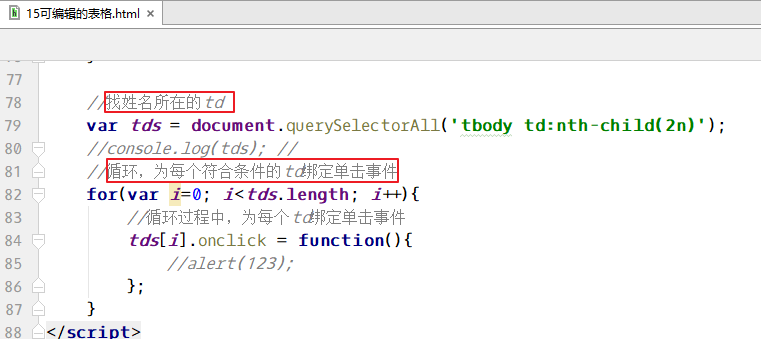
css代码：



2、JS做隔行换色



3、给姓名所在的td添加单击事件

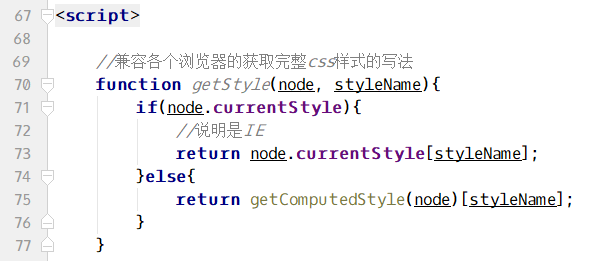


4、单击td，创建input并放到td中

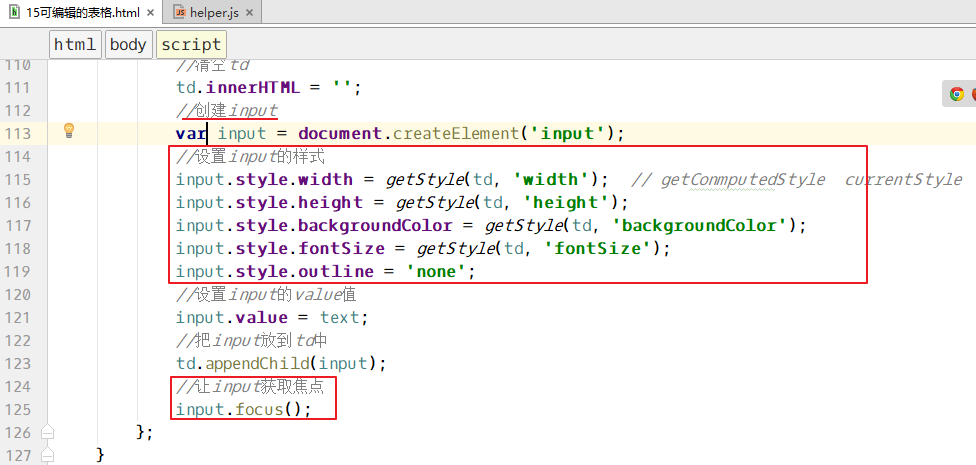


5、设置input的样式和td一样

复制前一天的getStyle函数（用于获取元素的样式）到当前的html中。



设置input的样式，并获取焦点：



6、给input绑定事件完全确定和取消功能

