事件基础2——内容

1. 事件流（Event flow）

1.1 什么是事件流

**DOM树**：浏览器首先会加载html，把html内容解析成一个DOM树，类似于我们把字符组成hash字符串解析成一个对象进行操作。

**Event**：当页面中的元素被操作的时候，浏览器会根据不同的行为触发一些不同的事件，比如：点击、鼠标移入、鼠标移出等。

**一个完整的事件过程**：捕获阶段——目标源——冒泡阶段

**事件流**：

当我们某个行为去触发了某个页面元素，比如点击的时候，告诉浏览器页面中某个元素需要触发一个事件，该事件会首先由外部进入到页面元素，这个过程叫做**事件捕获**。

因为DOM中每一个节点都会有一个父级（除了顶层节点以外），当某个DOM节点触发了事件以后，除了自己能触发该事件以外，同时他会把这个事件传递给他的父级，当一个节点接收到事件以后，会把该事件一级一级的通知给父级，这个过程我们称为**事件冒泡**，如果当前这个元素接收到了事件（无论是直接触发，还是子级通知的）只要我们给当前元素的当前事件绑定函数，那么这个函数就会被执行。

**注意1：**

当一个事件发生的时候，从window到目标元素每个元素会触发**两次**该事件，捕获阶段一次，冒泡阶段一次，所以我们可以给捕获阶段、冒泡阶段分别去绑定函数。

**注意2：**

通过on属性进行的绑定，绑定的是冒泡阶段

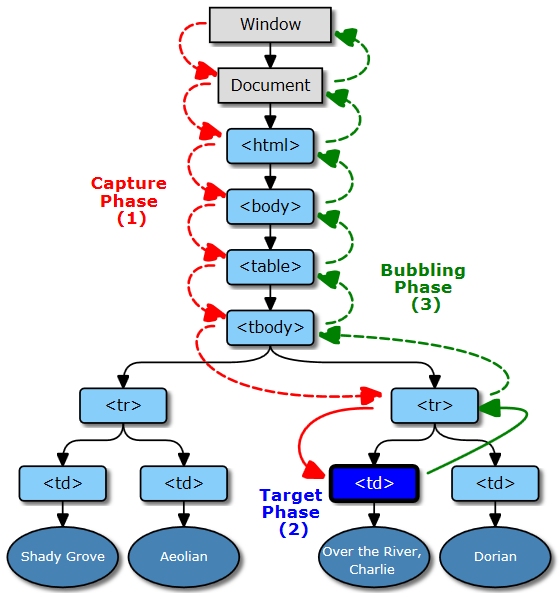
如果我们想绑定捕获阶段，那么on的方式实现不了，on的方式只能绑定事件的冒泡阶段，如果我们想绑定事件的两个阶段，则需要使用addEventListener方法。

addEventListener(事件名称，事件绑定函数，是否绑定捕获阶段);

第三个参数可以设置当前绑定的函数的阶段，默认false，表示绑定的是冒泡阶段；

不过一般情况下，我们很少需要用到捕获阶段；

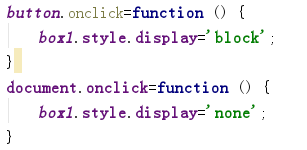
另外还有一定的兼容问题：不支持addEventListener的浏览器就没有这个捕获阶段的绑定。



2. 冒泡带来的问题，为什么要冒泡

2.1. 冒泡带来的问题

当我们点击button的时候，会把div显示出来，但是事件会冒泡到document，然后又隐藏起来了。

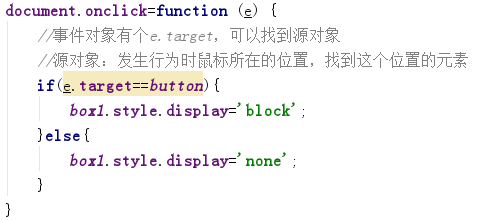


2.2 为什么要冒泡

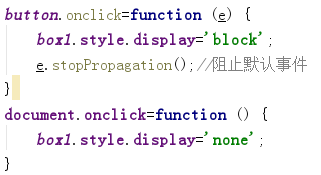
我们做的页面一般会有很多的其他元素，如果没有冒泡机制，我们就需要做每一件事情的时候对页面中每一个元素都要进行处理，有冒泡机制，其他的元素的点击就不用自己去做处理，把这个事情交给他们共同的父级。

**解决冒泡：**

**方法一：**存在问题，如果都写在document顶层对象中，页面就会比较乱；



**方法二：**js提供了阻止冒泡的方法



3. 阻止冒泡——方法的兼容性

冒泡默认是存在的，但是我们可以在需要的时候阻止他；

该方法可以阻止 当前对象的当前事件 的冒泡，换句话说，不同对象的不同事件的冒泡需要每一个一个的单独的去阻止。

**注意：**同一个对象不同事件的冒泡需要一个个单独去阻止。

**阻止冒泡的方法**：

event.stopPropagation()标准推荐的方法；

event.cancelBubble = true也可以阻止冒泡，但是标准不推荐，不过浏览器都支持。

例子：阻止了click的冒泡，没有阻止mouseover的冒泡，mouseover就会往上冒泡。



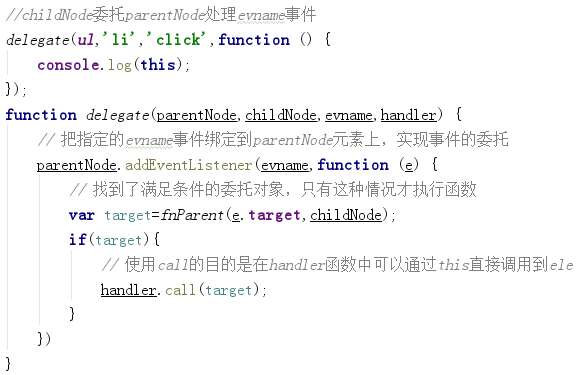
e.cacleBubble

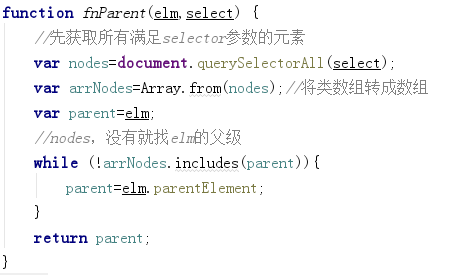
e.stopPropagtion()

4. 冒泡的应用——事件委托，事件委托的函数封装

利用事件冒泡的特性，把子级需要处理的任务委托给父级处理。

首先参考元素子级处理的任务是不是相识，相识的话就用事件委托来做。





5. 默认事件，阻止默认事件

5.1 存在的默认事件

当一个事件触发的时候，默认会做的一些事情

比如：选择文字、滚动页面、右键菜单、内容输入

5.2 阻止默认事件

两要素：找事件元素、事件源

找到该行为的事件源

在该事件的执行函数中使用: event.preventDefault() 来阻止默认行为

注意兼容：

return false是所有浏览器支持的方法；

event.preventDefault()标准推荐的方法；

**注意：**如果事件是通过addEventListener绑定的，那么只能使用event.preventDefault()方法来阻止，return false无效。



6.事件分类1——鼠标事件

**mousemove**

当鼠标在元素上移动的时候触发；

不是每移动一个像素触发，而是在一个极短的间隔时间内容，如果鼠标的位置发生了改变就会触发一次。

**mouseover——mouseout**

鼠标从元素外部移入元素内部的时候触发

**注意：**从子级移入到父级的时候，**也会触发**父级的mouseover,mouseout类似。

**mouseenter——mouseleave**

鼠标从元素外部移入元素内部的时候触发。

**注意：**从子级移入到父级的时候，**不会触发**父级的mouseenter,mouseleave类似

**mousedown——mouseup**

**click 、 dbclick**

click是由多个事件组合完成的一个事件，mousedown和mouseup；

当mousedown和mouseup在一个元素上触发的时候才会触发click；

当一个元素触发了click，其实触发的事件与顺序如下：

mousedown -> mouseup -> click

**mousewhile** 兼容性

鼠标滚轮事件

chrome/ie: **mousewheel**

firefox: **DOMMouseScroll** 该事件没有对应的on属性，需要使用addEventListener

滚轮方向

chrome/ie:**wheelDelta**；上+，下-

firefox:**detail**；上-，下+；

7.事件分类2——表单事件

**focus**

当元素获取到焦点的时候触发；

页面中的元素是有很多的，有些时候我们需要对某个元素进行操作，比如输入内容，为了能够把内容输入到正确想要的元素里面，在页面中设置叫做**焦点**概念，页面中如果有**一个元素**有焦点，那么表示该元素是激活的状态-**焦点状态**，那么内容输入就是针对该元素来进行的；

一个页面同时只能有一个元素被激活，当一个元素设置了焦点状态，那么就意味这有另外一个元素失去了焦点；

不是所有元素都有焦点（设置焦点），能够交互的元素才有焦点；

我们可以通过多种方式来设置一个元素的焦点：点击、tab、js，

**注意：**页面默认的焦点在**document**。

顺序：**focus——input——change——blur**

**blur**

当元素失去焦点的时候触发。

**input**

当元素的值发生改变的时候触发，每输入一个值，触发一次。

**change**

当前元素失去焦点的时候，如果内容发生了改变，则触发change。

**focus（）**

**获取焦点**

**blur（）**

**取消焦点**

**select（）**

8. 事件分类3——键盘事件

**keydown**

当按键按下的时候触发。

**keyup**

当按键抬起的时候触发。

**keypress**

当按键按下的时候触发，和keydown不一样的是不是所有按键都会触发keypress，比如功能键。

**注意：**

顺序：keydown——keypress——keyup

不是所有元素都支持键盘事件，支持焦点的元素才支持键盘事件，当按键触发的时候，当前具备焦点状态的元素才触发键盘事件。

9. 上下文菜单事件

**contextmenu**

上下文菜单-环境菜单，在不同的情况下会出现不同的菜单选项；

该事件不是右键事件；

因为这个菜单不仅仅可以通过右键的方式打开，还可以通过快捷键来调用；