正课：

1. 事件

1. 事件:

什么是: 浏览器自动触发的或用户手动触发的页面内容状态的改变。

比如: 单击按钮时，更改下拉列表中的选中项时...

什么是事件属性: 每个元素上都有一批以onxxx开头的特殊属性。当事件发生时，浏览器会找到发生事件的元素上的对应事件名称的onxxx属性，自动调用。

比如: btn.onclick:null

select.onchange:null

什么是事件处理函数: 提前绑定在onxxx属性上的函数，在事件发生会自动执行。这样的函数就叫事件处理函数。

何时: 今后，只要发生事件时，希望按照预先的约定，自动执行一项任务时。

如何:

提前绑定: 3种:

1. 在html中绑定:

<ANY on事件名="js语句">

比如: <button onclick="fun()">

js中: function fun(){ ... }

问题: 不符合内容与行为分离的原则，不便于维护

2. 在js中绑定用=赋值方式绑定:

ANY.on事件名=function(){

... this->触发事件的当前元素 ...

}

比如: btn.onclick=function(){ ... }

问题: 一个事件属性，只能绑定一个处理函数。无法在事件发生时，自动执行多个任务。

3. 在js中，用添加事件监听对象的方式

ANY.addEventListener("事件名",事件处理函数)

添加 事件 监听

比如: btn.addEventListener("click",function(){

... this->触发事件的当前元素 ...

})

优点: 一个元素的一个事件上可绑定多个处理函数。不想要，还可移除。

原理:

1. 浏览器内部其实有一个保存所有事件监听对象的队列。队列中保存了多个监听对象:

每个监听对象都记录了: 元素+事件+处理函数的一个组合。

2. 每调用一次addEventListener()，就会在监听对象队列中，添加一个监听对象，其中保存这件事相关的元素+事件+处理函数的组合。

比如: btnShoot.addEventListener(

"click", function(){ ... }

)

会创建一个监听对象:

{

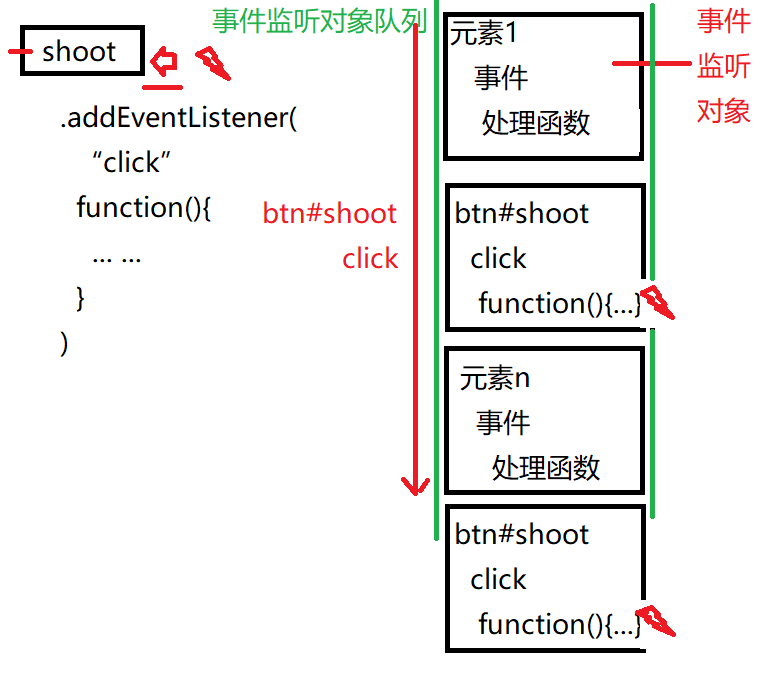
元素:btnShoot,

事件:click,

处理函数: function(){ ... }

}

3. 当事件发生时，浏览器以遍历监听对象队列的方式，查找符合条件的监听对象，自动调用其中的处理函数。比如: 当单击btnShoot按钮时，会用两个条件: 元素=btnShoot且事件=click，去队列中查找符合条件的监听对象。找到几个监听对象，就自动调用监听对象中的处理函数。



移除监听对象:

元素.removeEventListener("事件名",原处理函数对象)

比如: btn.removeEventListener("click",原处理函数对象)

坑: 移除监听对象时，要求找到原处理函数对象。但是因为匿名函数只用一次后，就找不到了！即使下次和上次写的函数表面上完全一样，已经不是同一个函数对象了。因为关键词function() 是创建新函数的意思。

所以:如果一个处理函数，有可能被移除时，就不要用匿名函数绑定。必须用有名字的函数绑定:

比如: btnShoot.addEventListener(

"click", shoot2

)

function shoot2(){ ... }

会创建一个监听对象:

{

元素:btnShoot,

事件:click,

处理函数:shoot2

}

当移除时:

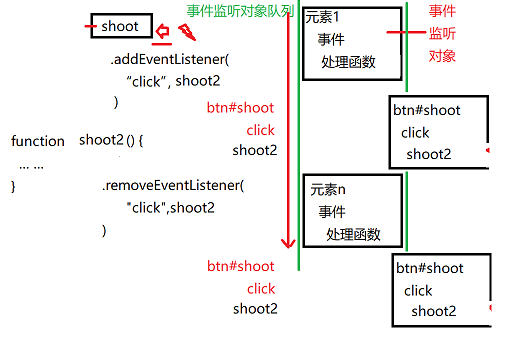
btnShoot.removeEventListener(

"click", shoot2

)

会用三个条件:

元素=btnShoot且事件=click且处理函数=shoot2，去监听对象队列中查找匹配的监听对象，并移除。



坑: 如果用有名称的函数绑定，则该事件处理函数，只能在一个元素的一个事件上绑定一次，不能重复绑定同一个处理函数多次！

\*\*\*\*\*事件模型: 鄙视常考！！！

什么是: 从事件发生时开始，一直到所有相关的处理函数被执行完，所经历的过程。

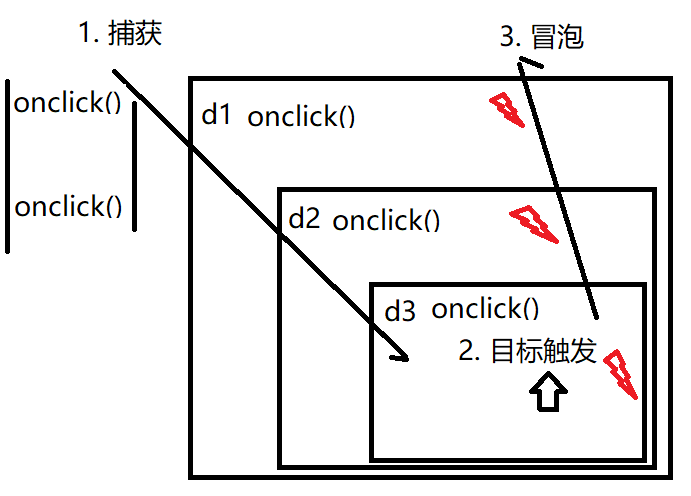
DOM标准的事件模型: 3个阶段

1. 捕获: 由外向内，记录各级父元素上的事件处理函数——仅记录不执行

2. 目标出发: 先触发目标元素上的处理函数

目标元素: 你本意想点击的那个元素

3. 冒泡执行: 由内向外，依次触发各级父元素上的处理函数。



事件对象:

什么是: 专门保存事件信息，并提供修改事件相关行为的函数 这样的对象就是事件对象

何时: 2种:

1. 获得事件相关的信息

2. 修改事件的相关行为

如何:

1. 不用自己创建事件对象。当事件发生时，浏览器自动创建一个事件对象，其中保存事件相关的信息和修改事件行为的函数。

2. 获取事件对象: 自动创建的事件对象，总是作为事件处理函数的第一个参数，自动传入处理函数中。

所以: 我们只要为事件处理函数准备一个形参e，就可以接住事件对象。并在处理函数中，使用e来获得事件的信息和修改事件行为。

比如:

事件发生时，自动创建event对象

↓

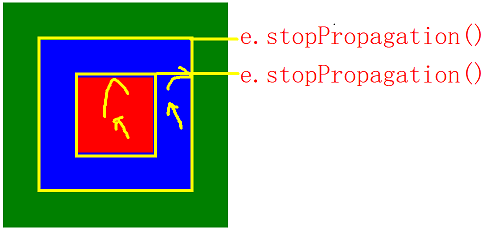
btn.onclick=function( e ){

//e->event对象

}

3. 用事件对象能做什么?

1. 取消冒泡: e.stopPropagation();



2. 事件委托/事件代理:

什么是事件委托: 将事件处理函数，统一定义在父元素上，请父元素代为保管。当子元素触发事件时，冒泡到父元素上，获得父元素上代为保管的处理函数使用。

为什么:

优化: 尽量减少事件监听的个数。

原因: 当事件发生时，浏览器是通过遍历监听队列的方式，找到符合条件的处理函数执行。遍历的速度，取决于元素的总数。元素总数少，遍历快！元素总数多，遍历慢！

如何: 今后只要多个子元素，都需要响应相同的事件时，只要将事件在父元素上绑定一次！所有子元素就可共用！只要任意一个子元素触发事件，都会冒泡到父元素上执行。

2大难题:

1. 事件处理函数中的this不再指向目标元素，而是指向父元素。——this不可用了！

解决: e.target

e.target专门用来在首次触发事件时，保存住目标元素，好处是不随冒泡改变而改变。

2. 不是所有e.target都是必须触发事件的。因为父元素下除了要触发事件的个别元素外，大部分元素可能是装饰性的。没有实际功能的。

解决: 在事件处理函数中，开始执行正常流程前，先验证是不是规定的元素。只有是规定的的元素，才执行正常流程。如果发现点的不是规定的元素，则什么也不做。

事件委托的另一个主要优点:

可让新增加的元素，不需要重复绑定事件，就可自动获得处理函数。

3. 阻止默认行为: e.preventDefault();

什么是默认行为: 有些元素自身带的行为。比如: <a href="#content1"> 当点a时，会在地址栏结尾添加锚点地址#content1，并且页面会上下晃动。

何止阻止默认行为: 只要一个元素自带的默认行为不是想要的，就可以阻止。

如何: e.preventDefault();

4. 鼠标坐标:

什么是: 当事件发生时，鼠标所在的位置。

何时: 只要根据鼠标的位置，来实现动画或交互效果时。

比如: 基于鼠标的游戏，飞机大战..

基于鼠标的交互效果，放大镜，拖拽

如何获得鼠标坐标: 3组:

1. 相对于屏幕左上角:

e.screenX, e.screenY

2. 相对于文档显示区左上角:

e.clientX, e.clientY

3. 相对于当前事件所在元素左上角:

e.offsetX, e.offsetY

5. 页面滚动效果:

什么是: 随着页面滚动到指定的位置，网页执行相应操作和变化。

比如: 天猫楼层效果，页头固定

如何:

1. 为窗口绑定滚动事件:

window.onscroll=function(){

}

2. 获得页面滚动过的距离：也就是页面顶部超出文档显示区上边沿的距离

随时根据滚动过的距离，决定页面发生何种变化。

如何: var scrollTop=document.documentElement.scrollTop||document.body.scrollTop

3. 主动修改页面滚动的位置：

window.scrollTo(横向滚动距离,纵向滚动距离)

window.scrollTo(0,0) 让页面滚动到左侧0位置，顶部0位置。

作业:

1. 将查询字符串转化为对象:

比如:

var search=

"?uname=dingding&upwd=123456&favs=swimming&favs=running&favs=music"

function search2Obj(str){

... ...

return obj;

}

var obj=search2Obj(search);

console.log(obj);

{

uname: "dingding",

upwd:"123456",

favs:["swimming","running","music"]

}

obj.upwd

2. 计算器:3\_calc.html

3. 3\_createTable.html

单词列表:

1. propagation: 蔓延

2. screen 屏幕

3. offset 偏移量

4. scroll 滚动