正课:

1. 事件

1. 事件:

什么是事件: 浏览器内容或元素状态的变化

什么是事件处理函数: 当事件发生时，自动调用的函数

何时: 如果希望事件发生时，能自动执行一项任务

如何绑定事件处理函数: 3种:

1. 在HTML中: <ANY on事件名="处理函数()"

js: function 处理函数(){ ... }

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，赋值方式:

elem.on事件名=处理函数;

function(){ ... }

问题: 一个事件只能绑定一个处理函数

3. 在js中绑定，添加事件监听对象的方式

elem.addEventListener("事件名",处理函数)

在elem元素的监听队列中添加一个监听"事件名"的对象，其中包含一个处理函数。

当elem元素上发生"事件名"时，会遍历监听队列，找到符合条件的处理函数，自动执行处理函数。

elem.removeEventListener("事件名",原处理函数)

强调: 如果一个处理函数有可能被移除时，则绑定时就要用有名称的函数作为处理函数。

移除时才能通过函数名找到绑定时使用的原函数

事件模型: 在发生事件时，浏览器的执行过程:

3个阶段:

1. 捕获: 由外向内，记录各级父元素上绑定的处理函数

2. 目标触发: 优先触发目标元素上的处理函数:

目标元素: 事件最初实际发生在的元素

3. 冒泡: 由内向外，依次触发捕获时记录的各级父元素上的处理函数

事件对象: 事件发生时，自动创建的，保存事件信息，并提供操作事件的API 的对象

何时: 获取事件信息，或修改事件的默认特性时

如何:

获取: 事件对象总是作为处理函数的第一个参数，默认传入

elem.on事件名=function(e){ e->event }

因为当事件发生时: elem.on事件名( event )

API:

1. 取消冒泡: e.stopPropagation();

//停止 蔓延

2. 利用冒泡:

优化: 尽量减少事件监听对象的个数

为什么: 浏览器触发事件是采用遍历的方式找到的监听对象。监听对象多，遍历就慢。

如何: 如果多个子元素需要绑定相同的事件处理函数时，其实只要在父元素上绑定一次即可。所有子元素共用。

两个难题:

1. 获得目标元素:

错误: this -> 父元素

正确: e.target 代替this

2. 判断目标元素是否想要的！

3. 阻止默认行为: e.preventDefault()

1. a当按钮用时: 如果href="#xxx", 会擅自修改地址栏中的url

2. 提交按钮: 也能实现自定义提交。

自定义提交: 2种:

1. input type=button

btn.onclick=function(){

先验证

如果都验证通过

才form.submit()

}

2. input type=submit

表单提交过程:

点submit按钮->触发form.onsubmit()->正式提交

先验证

如果有一项验证未通过

就阻止提交

3. HTML5的拖拽API: 可能和浏览器的一些快捷键发生冲突。通常都要先阻止默认的行为，再编写自己的拖拽。

4. 事件发生时，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的偏移量:

e.offsetX e.offsetY

5. 滚动事件:

//当窗口中的内容发生滚动时，自动执行

window.onscroll=function(){

//获得页面滚动过的距离: 页面顶部超出文档显示区的距离。

var scrollTop=document.body.scrollTop

||document.documentElement.scrollTop

}