正课:

1. navigator

2. 定时器

3. event

1. navigator

userAgent: 包含浏览器名称，内核，版本号的字符串

鄙视: 如何判断浏览器名称和版本号

2. 定时器: 2种:

1. 周期性定时器:

什么是: 让程序每隔一段时间间隔，反复执行一项任务

何时: 只要让程序每隔一段时间，反复执行一项任务

如何: 3件事:

1. 定义任务函数: 定时器每次反复调用的函数

2. 将任务函数，放入定时器中，启动定时器:

var timer=setInterval(task, 间隔ms)

timer是唯一标识该定时器的序号，专门用于停止定时器

3. 停止定时器: clearInterval(timer)



2种: 1. 手动停止:

2. 自动停止: 在任务函数中判断临界值，如果没有达到临界值，则继续执行。如果达到临界值，则自动调用clearInterval()

2. 一次性定时器:

什么是: 让程序先等待一段时间，再延迟执行一项任务。

执行完一次后，自动停止。

何时: 只要希望一段程序延迟执行时

如何: 3件事:

1. 定义任务函数: 定时器延迟执行的任务

2. 启动定时器: 启动等待

timer=setTimeout(task, 等待ms);

3. 停止定时器: 停止等待，不再执行

clearTimeout(timer);

定时器原理:

1. 启动定时器时，都不会立刻执行任务函数。仅是创建定时器对象，封装任务函数的定义。

2. 定时器执行时，不会立刻执行任务函数，而是将任务函数添加到回调队列中等待。

3. 只有主程序执行完，回调队列中的任务函数才会执行

结论: 定时器的任务函数，只能在主程序执行完才能执行

3. 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

事件处理函数: 当事件发生时，自动执行的函数

何时: 如果希望事件发生时，自动执行一项任务时

如何绑定: 3种:

1. 在HTML中: <ANY on事件名="js语句"

问题: 不符合内容与行为分离的原则，不便于维护

2. 在js中动态绑定: 赋值方式:

elem.on事件名=function(){

//this->当前elem

}

触发事件: elem.on事件名();

问题: 一个事件属性，只能绑定一个处理函数

3. 在js中动态绑定: 添加事件监听方式:

elem.addEventListener("事件名",handler)

elem.removeEventListener("事件名",handler)

强调: removeEventListener必须和addEventListener时写的内容完全一样。且，必须能够找到原处理函数对象。

所以: 如果一个处理函数可能被移除，则绑定时，就不能用匿名函数。必须用有名称的函数。

事件模型: 事件发生时经历的过程：3个阶段

1. 捕获: 由外向内，逐级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: 最初触发事件的第一个元素

3. 冒泡: 由内向外，按捕获阶段的反向，触发各级父元素上的处理函数。

为什么: 所有浏览器开发者，都认为，点在内层子元素上，也等效于点在外层元素上。

事件对象:

什么是: 事件发生时，自动创建的，封装事件信息的对象，并提供了操作事件行为的API。

何时: 1. 获取事件信息

2. 修改事件的默认行为

如何:

获取: DOM: 事件对象总是作为处理函数的第一个参数自动传入

elem.onclick=function(e){

//当事件发生时，e会自动获得事件对象

}

API:

1. 取消冒泡/停止蔓延:

e.stopPropagation()

2. 利用冒泡:

优化: 尽量减少事件监听的个数:

为什么: 浏览器触发事件处理函数是通过遍历的方式，查找所有监听对象。如果监听对象多，则遍历慢。

何时: 多个子元素需要绑定同一事件处理函数时

如何: 只需要在父元素上绑定一次处理函数即可

问题:

1. this不再指向目标元素，而是指向父元素

2. 鉴别目标元素

解决:

1. 不能用this, 应该用e.target

e.target始终保存着目标元素，不随冒泡而改变

2. 在事件处理函数中，先判断e.target的特征(标签名, class...)。只有目标元素满足要求，才能继续执行

不利用冒泡 vs 利用冒泡:

1. 绑定位置: 不利用冒泡 直接绑在每个子元素上

利用冒泡 绑在父元素上

2. 监听个数: 不利用冒泡 多

利用冒泡 少

3. 效率: 不利用冒泡 低

利用冒泡 高

4. 对动态添加的新元素:

不利用冒泡 不能自动获得事件处理函数

利用冒泡 可以自动获得父元素上的处理函数

3. 阻止默认行为:

为什么: 浏览器为了用户体验的考虑，可能增加了个别默认的行为。默认行为可能和我们添加的事件冲突

何时: 只要希望不要触发浏览器默认的行为时

典型场景:

1. 用a元素当按钮时

2. 用submit按钮也能实现自定义表单提交：

组合1:

<input type="button" onclick() form.submit()

组合2:

<input type="submit"

form.onsubmit()

e.preventDefault();

3. 使用HTML5中的拖拽API前，都要阻止默认行为

如何: e.preventDefault();