**正课:**

**1. 定时器:**

**一次性:**

**2. \*navigator:**

**3. \*\*\*event**

**1. 定时器:**

**一次性: 让程序先等待一段时间，再延迟执行一项任务**

**执行后，自动停止**

**何时: 只要让程序延迟执行一项任务，且不需要反复执行时**

**如何: 3步:**

**1. 任务函数:**

**2. 启动定时器: timer=setTimeout(任务函数,ms)**

**其中: ms指延迟的毫秒数**

**3. 停止定时器: clearTimeout(timer)**

**在执行前，停止等待，不再执行任务**

**鄙视: 定时器中的任务函数，必须等到主程序所有语句执行完，才能执行！**

**var a=10;**

**function fun(){**

**a++;**

**}**

**setTimeout(fun,0);**

**//fun无论等待多长时间都必须到最后才执行**

**console.log(a);//10**

**2. navigator:**

**什么是: 封装浏览器配置信息的对象**

**何时: 只要读取浏览器配置信息时**

**如何:**

**1. 判断是否启用cookie**

**什么是: 客户端持久存储用户私密信息的小文件**

**为什么: 内存中的数据都是临时的**

**何时: 只要在客户端持久保存数据时**

**如何判断是否启用cookie:**

**var bool=navigator.cookieEnabled**

**设置: 设置->高级->隐私->内容设置->查看和禁用cookie**

**2. 判断是否安装插件:**

**什么是: 为浏览器添加新功能的小软件**

**如何判断插件是否安装:**

**navigator.plugins["插件名"]!==undefined**

**3. 判断当前浏览器名称和版本号——鄙视**

**navigator.userAgent: 保存浏览器名称，版本，内核信息的字符串**

**何时: 只要判断浏览器名称和版本时**

**3. \*\*\*event: ——DOM**

**什么是: 用户手动触发的，或浏览器自动触发的页面状态的改变。**

**事件处理函数: 当事件发生时，自动执行的函数**

**绑定事件处理函数: 3种:**

**1. 在HTML中绑定事件处理函数:**

**<ANY ... on事件名="js语句" ...>**

**问题: 1. 不便于集中管理事件**

**2. 不便于灵活重用**

**总之: 不符合内容与行为分离的原则**

**2. 在js中，用赋值方式绑定:**

**ANY.on事件名=function(){**

**this->ANY 当前触发事件的.前的元素**

**}**

**问题: 赋值是替换原函数。每个事件只能绑定一个处理函数**

**3. 在js中，为元素添加事件监听对象:**

**ANY.addEventListener("事件名",handler)**

**优: 一个事件，可同时添加多个处理函数**

**可随时添加和移除**

**如何移除:**

**ANY.removeEventListener("事件名",原handler);**

**问题: 如果一个处理函数，可能被移除，则绑定时，就必须用有名的函数。不能用匿名函数。**

**事件模型: 当事件发生时，浏览器触发事件的过程——鄙视**

**DOM标准认为: 点在内层元素上，也等效于点在外层元素上了**

**3个阶段:**

**1. 捕获: 由外向内记录各级父元素绑定的处理函数**

**捕获阶段只记录处理函数，不执行**

**2. 目标触发:**

**目标元素: 最初实际触发事件的元素**

**优先触发目标元素上的处理函数**

**3. 冒泡: 由内向外，按捕获阶段顺序的反向，依次触发父元素上的处理函数**

**事件对象:e**

**什么是: 事件发生时，自动创建的记录事件信息的对象**

**何时: 只要获得事件的信息，或修改事件的默认行为**

**如何:**

**创建: 自动创建**

**获取: 事件对象e总是作为处理函数的第一个参数，自动传入。**

**API:**

**取消冒泡: e.stopPropagation();**

**利用冒泡:**

**优化: 减少事件监听的个数**

**为什么: 浏览器触发事件处理函数，是用遍历方式找打处理函数并执行。**

**何时: 只要多个平级子元素，要绑定相同事件时**

**如何: 只要在父元素上绑定依次处理函数，所有子元素自动共用！**

**2大难题:**

**1. 获取目标元素:**

**错误: this->父元素**

**正确: e.target->记录了最初触发事件的元素**

**且不随冒泡而改变！**

**2. 鉴别e.target是否是想要的:**

**元素名, class属性**

**阻止默认行为:**

**何时: 只要一个元素的事件中，带有默认行为，且默认行为不是想要的，就要阻止**

**如何: e.preventDefault();**

**三个典型:**

**1. <a href="#xxx"**

**默认: 跳到锚点，在url结尾增加#xxx**

**2. 阻止表单自动提交！**

**自定义表单提交: 2种:**

**1. <input type=button js: form.submit()**

**2. <input type=submit**

**=>form.onsubmit(): e.preventDefault()**

**3. HTML5中做拖拽效果时**

**必须阻止浏览器默认的拖拽行为**