正课:

1. BOM常用对象:

history location navigator screen

2. \*\*\*\*event

1. BOM常用对象  
history: 封装当前窗口打开后，成功访问过的url的历史记录栈

history.go(n)

固定套路: history.go(1) 前进

history.go(-1) 后退

history.go(0) 刷新

也可前进/后退多步

location: 封装当前窗口正在打开的url的对象

属性: .href 完整的url

不但可获取，还可设置，实现在当前页面打开新连接

.protocol: 协议

.host: 主机名+端口号

.hostname: 主机名

.port: 端口号

.pathname: 相对路径

.hash: 获得#锚点地址

.search: 获得表单提交后的查询字符串

方法:

跳转: 在当前窗口打开，可后退

location.assign("url")

=>location.href="url";

=> location="url";

在当前窗口打开，禁止后退

location.replace("url")

刷新: 2种:

浏览器缓存: 浏览器在首次加载页面时，会将变化不大的静态资源(css,图片,html)缓存在客户端本地硬盘。

为什么: 避免重复加载，节约网络流量，提高加载速度

加载页面时: 优先在本地找可用的资源，如果本地资源比服务器上资源旧，才从服务器下载新资源。

1. 普通刷新: 优先使用本地缓存中的文件

F5

history.go(0)

location.reload(/\*false\*/);

force

2. 强制刷新: location.reload(true)

force

navigator: 封装浏览器配置信息的对象:

cookieEnabled: bool值, 判断浏览器是否启用cookie

什么是: 客户端持久保存用户私密信息的小文件

为什么: 程序内存中的数据都是临时的

何时: 只要在客户端持久保存用户数据时

典型: 记住密码, 游戏最高分

plugins: 保存插件信息的集合

什么是: 为浏览器添加新功能的小软件

userAgent: 保存浏览器名称和版本号的字符串

何时: 鉴别浏览器的名称和版本号时

作业: 查找网上辨别浏览器名称和版本号的方法

screen: 专门保存显示设备信息

屏幕宽: screen.width 屏幕高: screen.height

2. \*\*\*\*event

什么是: 浏览器自动触发的，或用户手动触发的页面状态的改变

事件处理函数: 当事件发生时，自动触发的函数——回调

如何绑定事件处理函数: 3种:

1. 在HTML中绑定:

<ANY on事件名="js语句"

问题: 不符合内容与行为分离的原则，不便于维护和重用

2. 在JS中绑定:

1. 用on事件名:

elem.on事件名=function(){

this -> 当前触发事件的.前的元素对象elem

}

问题: 一个事件只能绑定一个事件处理函数

解决:

2. 用addEventListener:

elem.addEventListener("事件名",fn)

移除事件: elem.removeEventListener("事件名",fn)

总结: 如果一个事件只有一个处理函数，且不会被移除时，首选onXXX——简单

如果一个事件需要同时触发多个处理函数，或处理函数可能被移除时，首先addEventListener——灵活

其实addEventListener可以用匿名函数绑定

问题: 使用匿名函数绑定的事件监听无法移除

解决: 如果一个事件监听可能被移除，就必须使用有名的函数绑定

\*\*\*事件模型:

DOM: 3个节点

1. 捕获: 由外向内，逐级记录各级父元素已经绑定的事件处理函数

2. 目标触发: 优先触发目标元素上的事件处理函数

目标元素(target): 最初实际触发事件的元素

3. 冒泡: 由外向内，按捕获阶段记录的反向，逐级触发父元素上的事件处理函数

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

提供了操作事件的API

何时: 2种:

1. 获得事件信息

2. 修改事件的行为时

如何:

获取: 事件对象，默认作为事件处理函数的第一个参数传入。

function 事件处理函数(e)

//在触发事件时，参数e会自动收到事件对象

API:

取消冒泡: e.stopPropagation()

利用冒泡:

优化: 尽量少的绑定事件处理函数

为什么: 因为浏览器是用遍历方式查找事件监听并执行，所以事件监听的个数越多，导致页面响应速度慢

如何: 如果多个平级子元素绑定相同的事件时，仅需要在父元素绑定一次，所有子元素公用即可！

2个难题:

1. 如何获得最初触发事件的目标元素:

问题: this不再指实际单击的子元素

而是指向公共父元素

解决: e.target代替this

2. 目标元素一定是想要的吗?

问题: 可能点在子元素，也可能点在父元素

解决: 利用冒泡的事件处理函数中，都要先判断目标元素是否想要，再决定是否触发事件。

阻止默认行为: e.preventDefault()

何时: 只要默认行为不是想要的，就可取消。

如何: 2个固定场景:

1. 阻止a元素作为按钮时，向url末尾添加锚点地址

2. 配合submit按钮阻止表单提交

表单的onsubmit事件会在以任何方式提交表单时自动触发。是表单提交的最后一关。专门用于表单中所有验证，决定最终能否提交。

总结: 2中控制表单提交的方式：

1. type="button" 只有验证通过才form.submit()

2. type="submit"

只要验证未通过就e.preventDefault();

3. HTML5 拖拽API 也要用e.preventDefault()阻止浏览器自带的默认拖拽行为

事件坐标: 事件发生时，鼠标的坐标位置:

3组坐标系:

1. 相对于屏幕左上角的坐标: e.screenX/screenY

2. 相对于文档显示区左上角: e.clientX/clientY

3. 相对于父元素左上角: e.offsetX/offsetY

页面滚动:

事件: window.onscroll 在页面发生滚动时自动触发

属性: 滚动的距离: scrollTop

body的顶部，超出文档显示区顶部的距离

document.body.scrollTop