正课:

1. HTML DOM常用对象

2. 什么是BOM  
3. window

4. 打开和关闭窗口

5. history

6. location

7. navigator

8. 定时器

1. HTML DOM常用对象:

Form:

获取: var form=document.forms[i/id]

属性: .elements 获得表单中所有表单元素(input, button, textarea, select)的集合

.elements.length 获得表单中表单元素的个数

.length => .elements.length

方法: form.submit() 代替提交按钮手动提交表单

自定义表单提交:   
 <input type="button"

.onclick=function(){

先验证所有表单元素的值是否符合要求

只有所有表单都符合要求

才手动调用form.submit()

}

表单元素对象:

获取表单元素: var elem=form.elements[i/name/id]

如果表单元素有name属性: form.name

方法: elem.focus()

elem.blur()

2. 什么是BOM: Browser Object Model

BOM是专门操作浏览器窗口或软件的API——没有标准

何时: 操作浏览器窗口，或访问浏览器软件的信息时

包含:

window: history

location

navigator

document

screen

event

3. window对象:

2个角色:

1. 代替ES标准中的global充当全局作用域对象

2. 封装所有原生API的大的对象，包括:

ES+DOM+BOM

属性:

窗口大小: 2组:

1. 完整窗口大小: [window.]outerWidth/outerHeight

2. 文档显示区大小: [window.]innerWidth/innerHeight

4. 打开和关闭窗口

[window.]open() 打开一个新窗口

close() 关闭当前窗口

5. history:

什么是: 当前窗口中，保存当前窗口打开后成功访问过的url的历史记录栈

何时: 用程序实现前进和后退时

如何: history.go(i)

前进一步: history.go(1)

后退一步: history.go(-1)

后退两步: history.go(-2)

刷新: history.go(0)

6. location:

什么是: 保存地址栏中正在打开的url的信息的对象

还提供了操作页面跳转的API

何时: 2件事:

1. 获得当前url相关的信息

2. 页面跳转  
 如何:

属性: .href 获得当前完整的url信息

.protocol 获得url中的协议部分

.host 获得url中的主机名+端口号部分

.hostname 获得url中的主机名

.port 获得url中的端口号部分

.pathname 获得url中的相对路径部分

.hash 获得url中的#锚点地址部分

.search 获得url中的?查询字符串部分

方法:

1. 在当前窗口打开，可后退:

open("url","\_self")

location.href=url => location.assign("url")

2. 在当前窗口打开，进制后退:

location.replace("新url")

用"新url"代替history数组中当前旧的url，以此来阻止后退。

3. 刷新: 2种:

1. 普通刷新: 优先从本地浏览器缓存中获取资源，除非本地缓存没有或过期，才被迫从服务器重新下载

history.go(0)

F5

location.reload()

2. 强制刷新: 强制浏览器跳过本地缓存，总是直接从服务器下载新资源。

location.reload(true)

7. navigator:

什么是: 封装浏览器配置信息的对象

何时: 只要获取浏览器的配置信息

包括:

.cookieEnabled 判断当前浏览器是否启用cookie!

什么是cookie: 在客户端本地硬盘持久保存一个数据的小文件

.plugins 封装浏览器安装的插件信息的集合

什么是插件: 为浏览器添加新功能的小程序

何时: 判断浏览器是否安装了指定的插件:

navigator.plugins["完整插件名"]!==undefined 说明安装了

.userAgent 保存浏览器名称和版本号的字符串

何时: 只要精确判断浏览器的名称和版本号时

如何:

8. 定时器: —— 回顾:

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

如何: 3件事:

1. 先定义一个任务函数

2. 启动定时器: 将任务函数放入定时器中反复执行:

var n=setInterval(任务函数, 间隔ms)

强调:

1. 如果定时器可能被停止，则必须在启动是获得定时器的需要n。

2. 任务函数后，不要加()，因为不是立刻执行，且不止执行一次。

3. 停止定时器:

clearInterval(n) //停止n号定时器

一次性定时器:

什么是: 让程序先等待一段时间后，再延迟执行一项任务

如何: 3件事

1. 任务函数

2. 启动定时器: 开始等待，等待结束后，自动执行一次任务。任务执行后，自动停止。

var n=setTimeout(任务函数, 等待ms)

3. 关闭定时器:

clearTimeout(n)