正课:

1. HTML DOM常用对象
2. 什么是BOM
3. Window对象
4. 打开和关闭窗口
5. History
6. Location
7. Navigator
8. 定时器
9. HTML DOM常用对象:

Table

行分组管着行:

添加行:

删除行: 行分组.deleteRow(i)

强调: i 是行在行分组内的相对下标位置

矛盾:   
行分组.deleteRow(i) i 要求是行分组内的相对下标位置

每个行上都有一个rowIndex属性，记录着行在整个表中的下标位置

所以: 行分组.deleteRow(tr.rowIndex)

结果会删除tr的下一行

解决: 今后删除行，一定不要用行分组作为.前的主语

应该用table.deleteRow(tr.rowIndex)

获取行: 行分组.rows[i]

tr管着格:

添加格: var td=tr.insertCell(i)

通常都是末尾追加: tr.insertCell()

删除格: tr.deleteCell(i)

获取格: tr.cells[i]

Form: 代表页面上一个<form>元素

获得表单对象: var form=document.forms[i]

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

.elements.length 获得表单中表单元素的个数

.length = .elements.length

方法: form.submit() 手动提交表单

表单元素:

获取: form.elements[i/name]

更简化: 如果表单元素有name属性，则可直接写:

form.name

Form.username

方法: elem.focus() 自动获得焦点

elem.blur() 自动失去焦点

自定义表单提交

<input typy=”button”

.onclick=function(){

先验证所有表单元素的值是否符合要求

只有所有表单符合要求

才用submit提交

}

1. 什么是BOM: Browser Object Model

专门操作浏览器软件的API

问题: 没有标准！很多API已经废弃

包括:

window: 是包含所有ES,DOM,BOM原生API的全局对象

history: 掌管历史记录

location: 掌管地址栏

document: 掌管所有网页内容: DOM

navigator: 掌管浏览器的配置信息

screen: 掌管显示设备的信息

event: 掌管着事件相关操作

1. Window:

2个角色:

1. 代替ES中的global充当全局作用域对象
2. 封装所有ES,DOM和BOM的原生API

属性: 获得浏览器窗口大小: 2组:

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

用途: outerHeight<指定值, 可判断窗口是否最小化了

1. 文档显示区大小: window.innerWidth/innerHeight

文档显示区：浏览器窗口中，专门用于显示网页内容的区域

4. 打开和关闭窗口:

API: window.open()

window.close()

打开新链接: 4种:

1. 在当前窗口打开，可后退

Html: <a href=”url” target=”\_self”>

Js: open(“url”,”\_self”)

1. 在当前窗口打开，禁止后退

Js: Function open2{location.replace(“新url”)}

原理: 用新url代替history中现有的url

1. 在新窗口打开，可打开多个

Html: <a href=”url” target=”\_blank”>

-

Js: open(”url”, ”\_blank”)

1. 在新窗口打开，只能打开一个

Html: <a href=”url” target=”自定义窗口名”>

Js: open(“url”,”自定义窗口名”)

其实, <a>的target属性是新窗口的名字(name)

name是在内存中唯一标示一个窗口的名称

浏览器规定，相同名称的窗口，只能打开一个

新打开的同名窗口，会覆盖旧的窗口

预定义窗口名:

\_self: 自动获得当前窗口自己的名字给新窗口

\_blank: 不指定窗口名，让浏览器随机生成窗口名

5. history:

什么是: 保存当前窗口打开后，成功访问过的url的历史记录栈(数组)

控制着: 前进后退

如何: history.go(n)

history.go(1) 前进一步

history.go(-1) 后退一步

history.go(-2) 后退2步

history.go(0) 刷新

6. location:

什么是: 专门保存当前窗口正在打开的url

何时:1. 分段获得url中部分内容时

2. 页面跳转/刷新时

如何:

属性: .href 完整的url

.protocol 协议 http: https: ftp: …

.host 主机名+端口号

.hostname 主机名

.port 端口号

.pathname 相对路径

.hash #锚点地址

.search ?查询字符串

方法:

1. 在当前窗口打开，可后退:

location.href=”新url”

其实可简写为: location=”新url”

1. 在当前窗口打开，禁止后退:

location.replace(“新url”)

1. 刷新页面:
2. 普通刷新: 优先从浏览器本地缓存中获取资源，本地缓存中没有或过期，才被迫重新从服务器下载。

history.go(0)

location.reload()

F5

1. 强制刷新:

location.reload(true) 强迫: 始终从服务器下载新资源

7. navigator:

什么是: 专门保存浏览器配置信息的对象

包括:

1. .cookieEnabled: 判断是否启用了cookie

什么是cookie: 客户端本次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: 只要希望持久保存的数据，都应该放在cookie中

典型: 记住密码

1. .plugins: 包含所有插件的集合

何时: 判断是否安装某个插件:

如何: if(navigator.plugins[“插件名”]===undefined)

说明没装

else

说明装了

1. .userAgent: 保存浏览器名称和版本号的字符串

何时: 专门用于判断浏览器的名称和版本号

8. 定时器:

2种:

1. 周期性定时器: setInterval()

什么是: 让程序每隔一段时间间隔，反复执行一项任务

如何:

启动: var n=setInterval(function(){

任务

}, 间隔ms)

停止: clearInterval(n) //n是内存中定时器的唯一序号

1. 一次性定时器: setTimeout()

什么是: 让程序延迟执行一项任务，执行后自动停止定时器。

如何:

启动: var n=setTimeout(function(){

任务

}, 延迟ms)

停止: clearTimeout (n)