正课:

1. 添加/删除

2. HTML DOM常用对象

BOM

3. window

4. \*\*\*定时器

1. 添加/删除:

优化: 尽量减少操作DOM树的次数

如何: 2种:

1. 同时添加父元素和子元素时，要先在内存中将子元素添加到父元素中，最后再将父元素一次性添加到DOM树

2. 如果父元素已经在树上，要同时添加多个平级子元素时:

文档片段:

什么是: 内存中，临时存储一棵dom子树片段的存储空间

何时: 只要同时添加多个平级子元素时

如何: 3步:

1. 创建文档片段对象

var frag=document.createDocumentFragment();

2. 先将子元素添加到文档片段中

frag.appendChild(child)

3. 将文档片段一次性添加到DOM树指定父元素下

parent.appendChild(frag)

强调: frag将子元素送到dom树后，自动释放

2. HTML DOM常用对象:

什么是: 对常用的元素，提供了简化版的API

优: 简化

缺: 不是万能

Image: 创建: var img=new Image();

Select:

属性: .selectedIndex 快速获得当前选中项的下标位置

.value 当选中项有value属性时，会返回option的value

如果选中项没有value属性，则用内容代替

.options: 获得select下所有option元素对象的集合

options.length 获得选项的个数

.length => .options.length

固定套路: 清空所有option

sel.innerHTML="";

sel.length=0; =>sel.options.length=0;

方法: add(option) 代替 appendChild(option)

问题: 不支持frag

remove(i) 移除i位置的option

Option:

创建: var opt=new Option(text,value);

属性: .index .text .value

table:管着行分组:

创建: var thead=table.createTHead()

var tbody=table.createTBody();

var tfoot=table.createTFoot();

删除: table.deleteTHead()

table.deleteTFoot()

获取: tabel.tHead table.tFoot table.tBodies[i]

行分组:管着行:

创建: var tr=行分组.insertRow(i)

固定套路: 1. 在结尾追加一行: 行分组.insertRow()

2. 在开头插入一行: 行分组.insertRow(0)

删除: 行分组.deleteRow(i)

强调: 主语是行分组时，i要求是在行分组内的相对下标位置

获取: 行分组.rows 获得行分组内所有行的集合

行:管着格:

创建: var td=tr.insertCell(i)

固定套路: 末尾追加: tr.insertCell()

删除: tr.deleteCell(i)

获取: tr.cells

form:

获取: var form=document.forms[i/id/name];

属性: form.elements 获得表单中所有表单元素的值

强调: 表单元素包括: input select textarea button

form.elements.length 获得表单中表单元素的个数

form.length==> form.elements.length

固定套路: 获得结尾的按钮:

var btn=form.elements[form.length-n]

方法: form.submit() //在程序中手动提交表单

表单元素:

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

如果表单元素有name属性，则: form.name属性值

方法: elem.focus() 让当前表单获得焦点

elem.blur()

BOM: Browser Object Model

什么是: 专门操作浏览器窗口或软件的API

没有标准!

window: 2个角色:

1. 代替ES中的GLOBAL充当全局作用域对象

2. 保存所有内置，全局的API和对象

属性: 文档显示区大小: 浏览器窗口中专门显示网页的区域

/\*window.\*/innerWidth,

/\*window.\*/innerHeight

功能: 打开和关闭窗口:

打开窗口: /\*window.\*/open("url","target")

4种:

1. 在当前窗口打开，可后退

HTML: <a href="url" target="\_self">

js: open("url","\_self ")

2. 在当前窗口打开，禁止后退

js: location.replace("新url")

用新的url代替history中当前地址

3. 在新窗口打开，可打开多个

HTML: <a href="url" target="\_blank">

js: open("url","\_blank ")

4. 在新窗口打开，只能打开一个

HTML: <a href="url" target="自定义的窗口名">

js:open("url","自定义的窗口名")

原理: target属性其实是为新窗口指定内存中唯一的窗口名

浏览器规定，同一时刻，同名窗口只能打开一个

后打开的会覆盖先打开的

预定义name:

\_self 自动使用当前窗口的name

\_blank 不指定窗口名, 每打开一个窗口，浏览器会自动随机生成内部窗口名

关闭窗口: /\*window.\*/close();

history: 保存当前窗口打开后，成功访问过的url的历史记录栈

不允许修改history内容!

只能三个操作:

前进 history.go(1)，

后退 history.go(-1), history.go(-2)，

刷新 history.go(0)

location:

什么是: 保存当前窗口正在打开的url地址的对象

属性:

.href: 获取或设置完整的url地址

.protocol: 协议

.host: 主机名+端口号

.hostname: 主机名

.port: 端口号

.pathname: 相对路径

.search: 查询字符串参数

.hash: 锚点地址

鄙视: 将search转化为对象形式:

方法:

1. 在当前窗口打开，可后退:

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

2. 在当前窗口打开，禁止后退:

location.replace("新url")

3. 刷新:

普通刷新: 优先从缓存中获取资源，缓存没有或过期，才去服务器找。

f5

history.go(0)

location.reload()

强制刷新: 无论有没有缓存，都强制从服务器获取新资源！

location.reload(true)

4. \*\*\*定时器: 2种:

1. 周期性定时器:

什么是: 让程序每隔一个时间间隔，自动反复执行一项任务

何时: 只要让程序按照指定的时间间隔反复执行一项任务

——动画！

如何: 3步:

1. 定义任务函数: 变化一次的函数

function task(){...}

2. 将任务函数放入定时器中反复执行:

var timer=setInterval(task,ms)

启动定时器，让定时器每隔ms毫秒，自动反复执行task函数

其中: timer 指当前定时器唯一的序号

专门用于停止定时器之值

3. 停止定时器: clearInterval(timer)

2种: 1. 定时器自动停止:

在任务函数中，设定临界值

一旦达到临界值，就自动调用clearInterval

2. 手动停止定时器

2. 一次性定时器: