**正课:**

**1. 添加/删除**

**2. HTML DOM常用对象**

**3. 什么是BOM**

**4. window**

**5. 打开新链接**

**6. history**

**7. location**

**8. navigator**

**1. 添加/删除:**

**添加: 3步:**

**1. 先创建空元素对象:**

**var a=document.createElement("a")**

**<a></a>**

**2. 设置关键属性:**

**a.href="http://tmooc.cn"**

**a.innerHTML="go to tmooc"**

**<a href="http://tmooc.cn">go to tmooc</a>**

**3. 将新元素添加到DOM树上**

**3种:**

**parent.appendChild(a) 将a追加到parent下子元素的末尾**

**parent.insertBefore(a,child) 将a插入到child之前**

**parent.replaceChild(a,child) 用a替换child**

**优化: 尽量减少操作DOM树的次数**

**原因: 每操作一次DOM树，就会导致重排重绘——效率低**

**如何: 2种情况:**

**1. 如果同时添加父元素和子元素，应先在内存中将子元素加到父元素，最后再一次性将父元素添加到DOM树**

**2. 如果父元素已经在页面上，只是添加多个平级子元素，要用文档片段。**

**文档片段: 内存中临时保存多个子元素的虚拟父元素**

**何时: 只要添加多个平级子元素到DOM树**

**如何: 3步:**

**1. 创建文档片段对象:**

**var frag=document.createDocumentFragment()**

**2. 将子元素加入到文档片段:**

**frag.appendChild(child)**

**3. 将文档片段添加到DOM树**

**parent.appendChild(frag)**

**强调: frag将子元素添加到DOM树后，自动释放，不会成为页面上真实的元素**

**删除元素: parent.removeChild(child)**

**2. HTML DOM常用对象:**

**HTML DOM对部分复杂的元素的API进行了简化**

**Image: 代表页面上一个<img>元素**

**唯一的简化: var img= new Image();**

**Select/Option: 代表页面上一个<select>元素及其内部的<option>元素**

**Select:**

**属性: .selectedIndex 获得当前选中项的下标位置**

**.value 获得当前选中项的值:**

**如果选中的option有value，则返回value**

**如果选中的option没有value，则返回内容**

**.options 获得当前select下所有option的集合**

**.options.length 获得当前select下option的个数**

**.length => .options.length**

**方法: .add(opt) 问题: 不支持文档片段**

**.remove(i) 删除i位置的option**

**事件: .onchange=function(){ ... } 当选中项发生改变时**

**Option: 代表select下一个option元素:**

**创建: var opt=new Option(text,value)**

**Table/...**

**Table: 代表页面上一个<table>元素**

**table管着行分组:**

**创建: var 行分组=table.createTHead()**

**.createTBody()**

**.createTFoot()**

**删除: table.deleteTHead()**

**.deleteTFoot()**

**获取: table.tHead**

**.tFoot**

**.tBodies[i]**

**行分组管着行:**

**创建: var tr=行分组.insertRow(i)**

**在行分组内i位置插入一个新行**

**固定套路:**

**1. 在开头插入新行: var tr=行分组.insertRow(0)**

**2. 在结尾追加新行: var tr=行分组.insertRow()**

**删除: 行分组.deleteRow(i)**

**删除行分组中i位置的行**

**强调: i要求是行分组内的下标位置**

**问题: 无法自动获得要删除行相对于行分组内的下标位置**

**解决: tr.rowIndex 记录着tr在整个表中的下标位置**

**但是，不能用行分组.deleteRow(tr.rowIndex)**

**应该：table.deleteRow(tr.rowIndex)**

**获取: 行分组.rows[i]**

**table.rows[i]**

**行管着格:**

**创建: var td=tr.insertCell(i)**

**固定套路: 行末尾追加新格: tr.insertCell()**

**问题: insertCell只能创建td，不能创建th**

**删除: tr.deleteCell(i)**

**获取: tr.cells[i]**

**Form/...**

**Form代表页面上一个<form>元素**

**获取: var form=document.forms[i/id]**

**属性: .elements 获得form中的表单元素（input select textarea button）**

**.elements.length 获得form中表单元素的个数**

**.length => .elements.length**

**方法: form.submit() 手动调用提交表单**

**表单元素:**

**获取: var elem=form.elements[i/id/name]**

**如果表单元素上有name属性，则: form.name**

**方法: .focus() 让当前元素获得焦点**

**.blur() 让当前元素失去焦点**

**3. 什么是BOM:**

**Browser Object Model**

**什么是: 专门操作浏览器窗口的API**

**问题: 没有标准！**

**包括:**

**window: 包含了所有ES+DOM+BOM的API**

**location:**

**history:**

**document:**

**navigator:**

**screen:**

**event:**

**4. window对象:**

**2个角色:**

**1. 代替ES中的global，充当全局作用域对象**

**2. 封装ES+DOM+BOM的所有API**

**属性:**

**窗口大小: 2组:**

**1. 整个浏览器的窗口大小: outerWidth/outerHeight**

**2. 文档显示区大小: innerWidth/innerHeight**

**方法:**

**.open() 打开一个新窗口**

**.close() 关闭当前窗口**

**5. 打开新链接:**

**4种:**

**1. 在当前窗口打开新链接，可后退:**

**html: <a href="url" target="\_self">**

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

**2. 在当前窗口打开新链接，禁止后退:**

**js: location.replace("新url") 用新url代替history中旧url**

**3. 在新窗口打开新链接，可打开多个:**

**html: <a href="url" target="\_blank">**

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

**4. 在新窗口打开新链接，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")**

**原理: 浏览器中每个窗口在内存中都有一个唯一的name**

**可认为指定窗口名，如果不指定，浏览器就随机生成**

**浏览器规定，相同窗口名的窗口，只能打开一个，后打开的同名窗口会把先打开的覆盖掉**

**内置窗口名: \_self 用当前窗口自己的名字打开新窗口**

**\_blank 不指定窗口名，让浏览器随机分配**

**6. history对象:**

**history: 保存当前窗口打开后，成功访问过的url的历史记录栈(数组)**

**可用程序控制前进和后退:**

**history.go(1) 前进一步**

**history.go(-1) 后退一步**

**history.go(0) 刷新**

**7. location对象:**

**什么是: 保存当前窗口正在访问的url地址的对象**

**属性:**

**.href 完整url**

**.protocol 协议**

**.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. 在当前窗口打开新链接，可后退:**

**location.assign("新url")**

**=> location.href="新url"**

**2. 在当前窗口打开新连接，禁止后退:**

**location.replace("新url")**

**用新url代替history中当前旧url来禁止后退**

**3. 刷新: 鄙视**

**2种:**

**1. 普通刷新: 优先从浏览器本地缓存中获取资源，缓存中没有或过期，才从服务器下载新资源**

**history.go(0)**

**F5**

**location.reload(/\*false\*/) false: 不强制**

**2. 强制刷新: 跳过浏览器本地缓存，总是从服务端下载新资源**

**location.reload(true) true: 强制**

**8. navigator对象:**

**什么是: 保存浏览器配置信息的对象**

**何时: 想查看浏览器的配置信息时**

**包括:**

**1. .plugins: 保存浏览器已安装的插件信息**