正课:

1. 添加/删除:

2. HTML DOM 常用对象

3. 什么是BOM

4. 打开和关闭窗口

5. history

6. location

1. 添加/删除

3步:

1. 添加一个空元素

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

<a></a>

2. 定义元素的关键属性

a.innerHTML="文本";

a.href="url";

<a href="url">文本</a>

3. 将新元素添加到DOM树

parent.appendChild(a) 在parent下子元素末尾追加a

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

为什么: 只要修改DOM树，就会导致重排重绘

减少修改DOM树的次数，就是减少重排重绘的次数

网页加载过程:

html -> DOM Tree

↓

Render Tree -> layout -> paint

↑ 重排 重绘

css -> cssRules

如何: 2种:

1. 同时添加父元素和子元素:

先在内存中，将所有子元素添加到父元素上

最后，再一次性将父元素添加到DOM树上

2. 如果父元素已经在页面上了，添加多个平级子元素:

使用文档片段:

什么是: 内存中临时保存多个平级子元素的虚拟父元素

何时: 添加多个平级子元素

如何: 3步:

1. 创建文档片段:

var frag=document.createDocumentFragment();

2. 将子元素临时添加到frag中

frag.appendChild(child);

3. 将文档片段整体添加到DOM 树

parent.appendChild(frag)

删除: parent.removeChild(child)

child.parentNode.removeChild(child)

2. HTML DOM常用对象:

Image: 代表一个<img元素

var img=new Image();

Select: 代表一个<select 元素

属性: .selectedIndex 获得当前选中项的下标

.value 获得当前选中项的值

如果option没有value，就用innerHTML代替

.options 获得当前select下所有option对象的集合

.options.length 获得select下所有option的个数

.length => .options.length

固定用法: 清空select下所有option： .length=0

方法: .add(option) 不支持文档片段

.remove(i)

Option: 代表一个<option元素

创建: var opt=new Option(text,value);

属性:

Table: 管着行分组:

1. 创建行分组:

var thead=table.createTHead();

创建thead，并同时追加到table

.createTBody()

.createTFoot();

2. 删除行分组: table.deleteTHead()

.deleteTFoot();

3. 获取行分组: table.tHead

.tFoot

.tBodies[i]

行分组管着行:

1. 添加新行: var tr=行分组.insertRow(i);

在行分组中i位置，添加一个新行

固定用法:

1. 末尾追加: .insertRow()

2. 开头插入: .insertRow(0)

2. 删除现有行: 行分组.deleteRow(i)

强调: i必须是行分组内的相对下标位置。

问题: tr.rowIndex 可获得行下标，但是相对整个table中所有tr的下标位置。和行分组的deleteRow要求不一致

解决: 不要用行分组删除行，而用:

table.deleteRow(tr.rowIndex)

3. 获取行: 行分组.rows[i]

行管着格:

1. 添加格: var td= tr.insertCell(i)

2. 删除格: tr.deleteCell(i);

3. 获取格: tr.cells[i]

Form元素: 代表页面上一个<form元素

获取: var form=document.forms[i/id];

属性: .elements 获得所有表单元素: input select textarea button

.elements.length 获得表单元素的个数

.length => .elements.length

方法: .submit() 手动提交表单

如何手动提交表单:

1. 将提交按钮改成普通的按钮

2. 为提交按钮绑定单击事件

3. 在单击事件中，再次调用验证方法。只有验证通过，才手动调用form.submit()。

表单元素:

获取: var elem=form.elemens[i/id/name];

更简化: 如果表单元素有name属性: form.name

方法: elem.focus() 让当前表单元素获得焦点

elem.blur() 让当前表单元素失去焦点

3. 什么是BOM: Browser Object Model

专门操作浏览器窗口的API，没有标准

包括:

window 2个角色:

1. 代替Global充当全局作用域对象

2. 封装所有ES,DOM,BOM的API

包括:

history

location

navigator

document

screen

event

4. 打开和关闭窗口:

<a href="url" target="\_blank"

/\*window.\*/open("url","\_blank|\_self")

/\*window.\*/close();

限制只打开一个: open("url","自定义窗口名")

原理: 每个窗口在内存中都有唯一的name属性，唯一标识当前窗口

浏览器规定，同一name属性的窗口只能打开一个

后打开的会将先打开的同名窗口覆盖

如何设置自定义name属性:

<a href="url" target="自定义窗口名"

open("url","自定义窗口名")

预定义name:

\_self: 用当前窗口自己的name打开新窗口，新窗口覆盖当前旧窗口

\_blank: 不指定窗口的name。打开时，会随机分配窗口名。几乎不重复。

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。

API: history.go(n)

比如: 前进一步 history.go(1)

后退一步 history.go(-1)

如果后退一步不管用，可后退两步history.go(-2)

刷新 history.go(0)

6. location:

什么是: 代表当前窗口正在打开的url对象

属性:

.href 获取并修改当前窗口打开的url地址

在当前窗口打开新链接，可以后退:

location.href="新url"

.protocol: 协议

.host: 主机名+端口号

.hostname: 仅主机名

.port: 端口号

.pathname: 相对路径

.hash: #锚点地址

.search: ?查询字符串

鄙视: "?uname=zhangdong&upwd=123456"

转为对象: var obj={uname:zhangdong,upwd:123456}

obj.uname obj.upwd

API:

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

.replace("url") 用新url代替正在浏览的url——禁止后退

.reload() => 重新加载当前页面

2种刷新:

1. 普通刷新: 优先从本地缓存中获取文件，除非本地缓存没有或过期，才重新请求服务器

history.go(0)

F5

location.reload()

2. 强制刷新: 跳过本地缓存，始终请求服务器上的新文件

location.reload(true)

force

7. navigator: 保存浏览器配置信息的对象

cookieEnabled: 判断浏览器是否启用cookie

什么是cookie: 客户端本地硬盘持久存储用户私密信息的小文件

为什么: 内存中的数据都是临时的，一旦程序关闭或关机，内存释放，数据丢失。

何时: 只要希望在客户端持久存储私密数据时

plugins: 包含浏览器安装的插件信息的集合

什么是插件: 给浏览器添加新功能的小软件

如何判断: navigator.plugins["插件名"]!==undefined