正课:

1. 添加/删除

2. HTML DOM常用对象

BOM

1. 窗口大小:

2. 打开和关闭窗口

3. \*\*\*定时器

4. BOM常用对象

1. 添加/删除:

3步:

1. 创建空元素

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

<a></a>

2. 设置关键属性

a.innerHTML="go to tmooc";

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

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

问题: 元素必须出现在DOM树的指定父节点下，才能被浏览器识别到，并画出来。

3. 将新元素添加到DOM树的指定父元素下

末尾追加: parent.appendChild(新元素)

中间插入: parent.insertBefore(新元素, 现有元素)

替换: parent.replaceChild(新元素, 现有元素)

优化: 尽量减少操作DOM树的次数

为什么: 每修改一次DOM树，都会导致重新计算页面布局(layout)

页面加载过程:

html -> DOM Tree

↓

Render Tree -> \*\*\*layout\*\*\* -> paint

↑ 计算每个元素的

css -> COM 绝对位置

如何: 2种情况:

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，再将父元素一次性添加到DOM树

2. 同时添加多个平级子元素:

文档片段: 内存中，临时保存多个平级子元素的虚拟父元素

何时: 只要添加多个平级子元素时

如何: 3步:

1. 创建文档片段:

var frag=document.createDocumentFragment();

2. 将子元素临时放入文档片段:

frag的用法和普通父元素完全一样:

frag.appendChild(子元素)

3. 将文档片段一次性加入DOM树:

parent.appendChild(frag)

强调: frag是临时的虚拟父元素，添加到页面后，自动释放，不占用页面空间。

删除: parent.removeChild(child);

2. HTML DOM常用对象: 对常用的复杂的HTML元素操作进行的简化。

Image Select/Option Table/行分组/tr/td Form/表单元素

Image: 创建: var img=new Image();

Select:

属性:

.selectedIndex 获得当前选中项的下标位置

.options 获得select下所有的option元素的集合

.options.length 获得或设置select下option的个数

清空select下所有option:

select.innerHTML="";

select.options.length=0;

.length 等效于: .options.length

select.length=0;

.value 获得当前选中项的value属性

如果选中的option有value，则返回value

如果没有value，用innerHTML代替

方法:

.add(option) 代替appendChild

问题: 不支持frag参数, appendChild支持frag

.remove(i) 删除i位置的option

事件: .onchange 当选中项发生改变时自动触发

Option:

创建: var opt=new Option(text,value);

属性: .text 代替 .innerHTML

Table: 管着行分组:

创建行分组: var thead=table.createTHead();

var tbody= .createTBody()

var tfoot= .createTFoot()

删除行分组: table.deleteTHead()

.deleteTFoot()

获取行分组: table.tHead

.tBodies[i]

.tFoot

行分组: 管着行:

添加新行: var tr=行分组.insertRow(i)

在行分组的i位置插入新行

原i位置的行，向后挤压，顺移

固定套路: 1. 末尾追加: 行分组.insertRow()

2. 开头插入: 行分组.insertRow(0)

删除行: 行分组.deleteRow(i)

获取行: 行分组.rows 返回行分组内所有行对象的集合

行: 管着格:

添加新格: var td=tr.insertCell(i)

用法和insertRow几乎完全一样

问题: 只能创建td,不能创建th

解决: css装饰thead中的td

删除格: tr.deleteCell(i)

获取所有格: tr.cells 返回行中的所有格对象

删除行:

不好的做法: 行分组.deleteRow(i)

问题: i是行在行分组内的相对下标位置，无法自动获得！

解决: 每个tr，都有一个rowIndex属性，记录了行在整个表中的下标位置——可自动获得！

好的做法: table.deleteRow(tr.rowIndex)

练习: 总结: 三大对话框:

alert() 警告

prompt() 输入

confirm() 确认 2个按钮: 确认/取消

返回值: 确认->true 取消->false

Form: 代表页面上一个表单元素

获取: var form=document.forms[i/id];

属性:

.elements 获得所有表单元素(input, select, button, textarea)

.elements.length 获取表单中表单元素的个数

.length 等效于.elements.length

方法: form.submit() 手动提交表单！

想手动控制是否提交表单时:

<input type="button" 配合 form.submit()

事件: .onsubmit 当表单提交之前自动触发

Element: 表单元素:

获取: var elem=form.elements[i/name/id]

更简化: 如果表单元素有name属性: form.name

固定套路: 获取表单中倒数第二个位置的提交按钮

form.elements[form.length-2]

方法: elem.focus(); 让当前表单元素获得焦点

elem.blur(); 让当前元素失去焦点

BOM: Browser Object Model

什么是: 专门操作浏览器窗口的API

问题: 没有标准！

何时: 要操作浏览器窗口时或要获取浏览器软件的配置信息

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

说明: window下的变量或函数，不用加window.也可直接使用。因为只要无主的变量和函数，默认都在window中找。

history: 封装当前窗口打开后成功访问过的url的历史记录栈

location: 封装当前打开的url对象

navigator: 封装浏览器的配置信息

document: 就是DOM中的树根节点document

screen: 封装当前显示设备的信息

event: 封装事件信息

1. 窗口大小:

窗口的完整大小：window.outerWidth/outerHeight

仅文档显示区大小: window.innerWidth/innerHeight——用的多

2. 打开窗口:

1. 在当前窗口打开，可后退

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

js: window.open("url","\_self");

2. 在当前窗口打开，不可后退

js: location.replace("url")

history: 专门保存当前窗口打开后，成功访问过的url的历史记录栈——数组

控制着前进和后退:

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

location: 专门保存当前正在打开的url的对象

location.replace("新url") 用新"url"替换当前正在打开的url

3. 在新窗口打开，可打开多个

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

js: window.open("url","\_blank");

4. 在新窗口打开，只能打开一个

html: <a href="url" target="自定义窗口名">xxx</a>

js: window.open("url","自定义窗口名");

其实: 每个窗口都有一个name属性

在内存中唯一标识一个窗口

浏览器规定，相同name属性的窗口，只能打开1个

后打开的会把先打开的替换

其实: target属性的值，应该是一个窗口名name

target分三种情况:

1. 自定义窗口名: 新打开的窗口就会代替先打开的旧窗口，保持只有一个同名窗口存在。

2. \_self: 预定义窗口名: 自动获取当前窗口名

3. \_blank: 预定义窗口名: 不指定任何窗口名

如果不指定窗口名，则浏览器会随机分配窗口名