正课:

1. HTML DOM常用对象:

Image Select/Option Table/... Form/...

2. BOM:

打开和关闭窗口:

窗口大小:

\*\*\*定时器:

BOM常用对象:

history location navigator screen

1. HTML DOM常用对象:

HTML DOM在创建和删除元素方面，对常用的复杂HTML元素也进行了简化

Image: 创建: var img=new Image();

Select:

属性:

.value 获得select元素选中项的value

如果选中项没有value属性，则用内容代替

.selectedIndex 当前选中项的下标

.options 获得select下所有option元素的集合

.options.length 表示选项的个数

不但可获取个数，还可清空所有选项:

.options.length=0

.length => .options.length

清空select,还可写为: select.length=0

方法:

.add(option) 追加一个option

问题: 不支持frag

vs appendChild 支持frag

.remove(i) 移除i位置的选项

事件: .onchange 当选中项改变时触发

Option:

创建: var opt=new Option(text,value);

Table: 管着行分组

创建行分组:

var thead=table.createTHead()

var tbody=table.createTBody()

var tfoot=table.createTFoot()

删除行分组:

table.deleteTHead()

table.deleteTFoot()

获取行分组:

table.tHead

table.tBodies[i]

table.tFoot

行分组: 管着行

创建行: var tr=行分组.insertRow(i)

固定套路: 1. 开头插入新行: 行分组.insertRow(0)

2. 末尾追加: 行分组.insertRow()

删除行: 行分组.deleteRow(i)

强调: i是要删除的行在行分组内的相对位置

获取所有行: 行分组.rows

行: 管着格

创建格: var td=tr.insertCell(i)

强调: insertCell只能创建td

固定套路: 末尾追加: tr.insertCell()

删除格: tr.deleteCell(i)

获取所有格: tr.cells

删除行: 2种:

1. 行分组.deleteRow(i) i是在行分组内的相对下标位置

问题: 没有办法自动获得行分组内的相对下标位置

2. table.deleteRow(i) i是在整个table中的绝对下标位置

如何自动获得行在整个表格中的位置: tr.rowIndex

总结: 今后只要删除行，都要首选用table删除:

固定套路: table.deleteRow(tr.rowIndex)

练习中: 三大对话框:

alert() 警告框

prompt() 输入框

confirm() 确认框: 有确认和取消操作的对话框

何时: 只要执行危险的操作前(提交更新、删除)都要先确认再执行操作

如何: var bool=confirm("提示消息");

如果点确定，就返回true，就执行操作

否则，就返回false，就不执行任何操作

form:

获取: document.forms[i/id]

属性:

.elements 获得表单中的所有表单元素

包括: input select button textarea

.elements.length 表示表单中表单元素的个数

.length => elements.length

方法: form.submit(); 用程序手动提交表单

何时: 只要希望通过自定义的验证决定是否提交表单

事件: form.onsubmit...

表单中的表单元素:

获得表单元素:

一般: var elem=form.elements[i/id/name]

简写: 前提: 必须有name属性的元素:

form.name

特例: 没有name属性的元素:

固定套路: 获得倒数第二个按钮:

form.elements[form.length-2]

方法: elem.focus() 让elem获得焦点

elem.blur() 让elem失去焦点

2. BOM:

什么是: 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ES中的Global充当全局作用域对象

2. 封装所有浏览器内置的/DOM/BOM的API

history: 专门保存历史记录,控制前进后退

location: 专门保存浏览器正在打开的url地址

navigator: 专门保存浏览器的配置信息

document: 就是DOM中的document节点

screen: 专门保存显示设备的信息

event: 专门保存事件信息

window:

打开和关闭窗口: 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，不可后退

js: location.replace("url")

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name">

js: open("url"," 自定义name ")

内存中每个窗口都有一个唯一的name属性

浏览器规定: 同一个name属性的窗口只能打开一个

后打开的同名窗口会覆盖先打开的

其实target属性值，就是窗口的name属性

预定义的name:

\_self: 用当前窗口自己的name打开新窗口

\_blank: 用空name打开新窗口

浏览器会自动为每个空name属性的窗口随机分配name属性

窗口大小:

文档显示区的大小:

宽: innerWidth 高: innerHeight

\*\*\*\*定时器:

周期性定时器:

什么是: 让程序每隔一段时间间隔反复执行一项任务

何时: 只要让程序每隔一段时间间隔反复执行一项任务

如何: 3件事:

1. 任务函数: function task(){... ...}

2. 启动定时器: var timer=setInterval(task, 间隔ms)

timer: 定时器在内存中的序号: 从1开始的整数

何时: 只要一个定时器可能被停止，都要在启动是先保存住序号

3. 停止定时器: clearInterval(timer)

问题: 停止定时器不会自动清空timer中的序号

解决: 建议停止定时器后，手动清空timer

timer=null;

2种停止方式:

1. 用户手动停止:

2. 自动停止:

在任务函数内反复判断临界值

只要达到临界值，就自动停止

一次性定时器:

什么是: 让程序先等待一段时间，再执行一次任务

执行后自动停止

何时: 只要让程序先等待一段时间，再自动执行一次任务

如何: 3件事:

1. 任务函数task

2. 启动定时器: var timer=setTimeout(task, 等待ms)

3. 停止定时器: clearTimerout(timer)

一般用于在执行任务之前，取消等待，不再执行