DOM Day01

1. DOM 概述：

JavaScript=ECMAScript(核心语法)+

DOM(专门操作网页的 API)——3 天

BOM(专门操作浏览器窗口的 API)——2 天

DHTML：所有实现网页动态效果技术的统称

DHTML：HTML+CSS+JavaScript

鄙视题：HTML XHTML DHTML XML

HTML: 超文本标记语言，专门定义网页内容的语言

XHTML: 严格的 HTML 标准

DHTML: 所有实现网页动态效果技术的统称

XML: 可扩展的标记语言，标签都是自定义的

XML 语法和 HTML 语法完全相同！——结构化数据

```
<Student id="1001">
 <sname>范冰冰</sname>
 <shx>91</shx>
</Student>
```

"{"id":"1001","sname":"范冰冰","shx":91}"——JSON

DOM：操作一切结构化文档的通用 API

用 DOM 即可操作 HTML，又可操作 XML

核心 DOM：可以操作所有文档的通用 API

API 繁琐！ 却可实现所有功能：增，删，修，查

HTML DOM：基于核心 DOM 的 API，扩展的专门操作 HTML 内容的 API

简单易用！ 重点简化了修改操作

对个别复杂 HTML 元素扩展了简化方法。

开发中：都是修改时，能用 HTML DOM，优先使用 HTML DOM

查，增，删时，只能用核心 DOM

2. ***DOM Tree：

网页加载进内存时，会先创建一个 document 对象，指代当前网页

网页中的所有内容:元素，文本，属性，注释等，都叫节点对象

Node

document 作为 DOM Tree 的根节点，所有节点对象都是 document 对象的子节点。

节点对象的类型：

document 对象：文档类型(Document)

元素节点：元素类型(Element)

文本节点：文本类型(Text)

属性节点：属性类型(Attr)

Node 类型：是所有节点的父类型

3 个通用属性：

nodeType: 获得当前节点的类型，返回的是一个数字

ELEMENT_NODE: 1

TEXT_NODE: 3

ATTRIBUTE_NODE: 2

DOCUMENT_NODE: 9

何时使用：专门判断不同节点类型

nodeName: 获得当前节点的名称：

如果是元素节点：返回*全大写字母*标签名

文本节点：返回#text

文档节点：返回#document

何时使用：专门用于区分不同的元素节点

比如：判断一个元素是不是按钮

if(elem.nodeName=="BUTTON")

nodeValue: 获得当前节点的值，对元素节点无效

如果是元素节点：返回 null！

文本节点：返回文本的内容

DOM Tree：6 种关系

1.父子关系：parentNode childNodes firstChild lastChild

childNodes: NodeList 集合——类数组对象

***动态集合：不包含结果对象的完整属性，仅保存对象的引用

每次使用，都重复查询！

```
for(var i=0,len=elem.childNodes.length;i<len;i++)
```

2.兄弟关系：previousSibling nextSibling

除 parentNode 外，都会受到看不见的空字符干扰！

*****递归遍历节点树：（手写！）

算法：深度优先遍历！

优先遍历下级节点，直到碰到叶子节点，才返回并更换另一个分支继续遍历下级节点。

递归：函数自己内部又调用了自己。

1. ***DOM Tree:

遍历节点树：深度优先算法，递归调用

节点树：包含所有节点对象的树结构。

arguments.callee:指代当前函数对象

何时使用：多用于在递归调用内部取代当前函数名

今后，只要递归调用，内部调用当前函数都要用 callee

遍历元素树：仅包含元素节点的树结构

仅是节点树的子集。

有兼容性问题

	节点树	vs	元素树
父节点	parentNode		parentElementNode
所有子节点	childNodes		children
第一个子节点	firstChild		firstElementChild
最后一个子节	lastChild		lastElementChild
前一个兄弟	previousSibling		previousElementSibling
后一个兄弟	nextSibling		nextElementSibling

遍历 API：2 个：NodeIterator TreeWalker

内部同样使用深度优先算法

2 步：

1. 创建迭代器对象：

```
var iterator=document.createNodeIterator(
        开始节点对象,显示何种节点,
    //NodeFilter.SHOW_ALL//NodeFilter.SHOW_ELEMENT
        null,false
```

);

*迭代器开始时，站在第一个节点的前一个位置！

两个方法：var nextNode=iterator.nextNode();

让迭代器向下一个对象跳一步

同时返回跳到的对象

如果没有下一个节点了，则返回 null

iterator.previousNode();

2. 利用循环推动迭代器反复向下一个节点移动

TreeWalker vs NodeIterator

创建和深度遍历的方法，完全一样

区别 1：TreeWalker 一开始就站在开始节点上

Iterator 一开始站子开始节点之前的空位置

区别 2：扩展了更灵活的跳转方法

parentNode()，firstChild()，nextSibling()...

3. *查找元素节点：2 大类：

1. 按 HTML 属性查找：

var elem=document.getElementById("id 值");

var elems=parent.getElementsByTagName("标签名");

不仅查找直接子元素，且同时查找所有子代元素

var elems=parent.getElementsByName("按 name 属性");

*****以上没有兼容性问题*****

*****下方 API IE8 不兼容*****

var elems=parent.getElementsByClassName("按 class 名");

***getElementsxxx，返回 HTMLCollection: 动态集合

for(var i=0,len=elems.length;i<len;i++)

课堂练习：

在 html 中绑定的事件处理函数，如要获得当前触发事件的元素对象，都要在 html 中显式用 this 关键字传入。

***不能再函数中直接用 this。

因为此时函数内的 this 相当于 window

何时使用 this：事件处理函数中用到当前触发事件的元素

2. Selector API: 专门利用 css 选择器查找任意元素的方法

jquery 的核心！

var elem=parent.querySelector("选择器");

只返回第一个匹配的元素对象

何时使用：只找一个元素时，使用！

var elems=parent.querySelectorAll("选择器");

***elems，返回所有符合选择器的元素对象

是包含完整对象属性的集合！——非动态集合

for(var i=0;i<elems.length;i++)

鄙视题：

Selector API vs getElementxxx

效率：getElementxxx 效率最高！

易用：Selector API 更容易使用，便于反复查找！

如果只找一次就可获得元素时，用 getElementxxx

如果需要反复查找才能获得元素时，用 Selector API

案例

<title>1. 使用节点树方式实现表格偶数行变色</title>

```
<style>.blue {background-color: #ccccff;}</style>
<script>
    window.onload=function(){
        //按 id 找到 table 对象
        var table=document.getElementById("data");
        //找到 table 对象下的 tbody
        var tbody=table.getElementsByTagName("tbody")[0];
        //找到 tbody 下所有 tr 对象
        var trs=tbody.getElementsByTagName("tr");
        /*遍历 trs 中所有行对象，
          i 从 0 开始，到<trs.length 结束,每次增 1*/
        for(var i=0,len=trs.length;i<len;i++){
            //判断 i 是否偶数行
            if(i%2!=0){//将当前 tr 的 className 改为 blue
                trs[i].className="blue";
            }
        }
    }
</script>
```

<title>使用 DOM 方法递归遍历节点树</title>

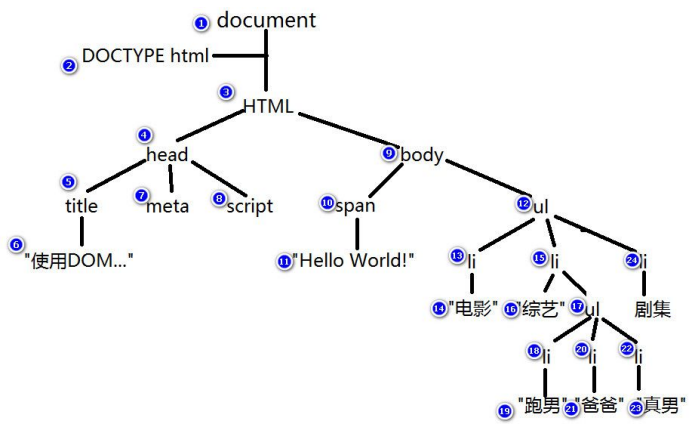
```
    <body>
        <span class="msg">Hello World!</span>
        <ul>
            <li>电影</li>
            <li>综艺
                <ul>
                    <li>跑男</li>
                    <li>爸爸</li>
                    <li>真男</li>
                </ul>
            </li>
            <li>剧集</li>
        </ul>
    </body>
//实现一个函数: 遍历指定节点下的所有子节点
window.onload=function(){
    var tabs=["|-"];
    /*遍历元素树*/
    (function(curr){
        console.log(tabs.join("")+curr.nodeName);
        //如果 curr 有子节点，就遍历 curr 下所有子节点
        var children=curr.children;
        if(children){//开始打印子元素之前，压入一个缩进
            tabs.unshift("\t");
            for(var i=0,len=children.length;i<len;i++){
                //将当前子节点，再作为父节点向下遍历
                //递归调用当前函数自己，传入新对象参数
                arguments.callee(children[i]);
            }//当前节点下的所有子节点输出完之后，
            tabs.shift();//弹出一个缩进
        }
```

```
    })(document);
    /*遍历节点树
    (function(curr){//如果 curr 不是元素类型或文档类型，
        //就输出 curr 的节点值，否则输出 curr 的节点名
        console.log(tabs.join("")+
(curr.nodeType!=1&&curr.nodeType!=9&&curr.nodeType!=10?
                                curr.nodeValue:curr.nodeName)
        );
        //如果 curr 有子节点，就遍历 curr 下所有子节点
        var children=curr.childNodes;
        if(children){//开始打印子元素之前，压入一个缩进
            tabs.unshift("\t");
            for(var i=0,len=children.length;i<len;i++){
                //将当前子节点，再作为父节点向下遍历
                //递归调用当前函数自己，传入新对象参数
                arguments.callee(children[i]);
            }//当前节点下的所有子节点输出完之后，
            tabs.shift();//弹出一个缩进
        }
    })(document);
}
<title>示例 1-使用遍历 API 遍历节点或元素树</title>
    <body>
        <span class="msg">Hello World !</span>
        <ul>
            <li>综艺</li>
            <li>电影
                <ul>
                    <li>捉妖记</li>
                    <li>煎饼侠</li>
                    <li>道士下山</li>
                </ul>
            </li>
            <li>剧集</li>
        </ul>
    </body>
window.onload=function(){
    var tabs=["|-"];
    //Step1: 创建迭代器对象：
    var iterator=document.createTreeWalker(
                        document,//开始节点
                    NodeFilter.SHOW_ELEMENT,
                        null,false);
    var node=iterator.lastChild();
    node=iterator.lastChild();
    node=iterator.firstChild();
    node=iterator.nextSibling();
    node=iterator.lastChild();
    node=iterator.previousSibling();
    console.log(node);
    /*Step2: 使用循环，推动迭代器跳向下一个节点*/
    var node;
    while((node=iterator.nextNode())!=null){
    console.log(tabs.join("")+(node.nodeType!=1&&node.nodeT
ype!=9&&node.nodeType!=10?node.nodeValue:node.nodeName)
        );
    if(node.childNodes.length!=0){//如果当前节点有子节点
            tabs.unshift("\t"); //压入缩进
        }else{//如果没有子节点，才考虑退格
            //下一个节点不是自己的兄弟
            var next=iterator.nextNode();
            if(next==null){break;//已经是最后节点，就退出
            }else if(node.nextSibling!=next){
                tabs.shift();
            }iterator.previousNode();
        }
    }
}
<title>实现购物车客户端计算</title>
<meta charset="utf-8" />
<style>
    table{width:600px; text-align:center;
        border-collapse:collapse;}
    td,th{border:1px solid black}
    td[colspan="3"]{text-align:right;}
</style>
<script src="js/6_2.js"></script>
</head>
<body>
    <table id="data">
        <thead>
            <tr>
                <th>商品名称</th>
                <th>单价</th>
                <th>数量</th>
                <th>小计</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>iPhone6</td>
                <td>&yen;4488</td>
                <!--￥-->
<td>
    <button onclick="calc(this)">-</button>
    <span>1</span>
    <button onclick="calc(this)">+</button>
</td>
                <td>&yen;4488</td>
```

```
                </tr>
                <tr>
                    <td>iPhone6 plus</td>
                    <td>&yen;5288</td>
                    <td>
                    <button onclick="calc(this)">-</button>
                    <span>1</span>
                    <button onclick="calc(this)">+</button>
                    </td>
                    <td>&yen;5288</td>
                </tr>
                <tr>
                    <td>iPad Air 2</td>
                    <td>&yen;4288</td>
<td>
    <button onclick="calc(this)">-</button>
     <span>1</span>
     <button onclick="calc(this)">+</button>
</td>
                    <td>&yen;4288</td>
                </tr>
            </tbody>
            <tfoot>
                <tr>
                    <td colspan="3">Total: </td>
                    <td>&yen;14064</td>
                </tr>
            </tfoot>
    </table>
</body>
```

<title>1. 实现伸缩二级菜单</title>

```
<body>
    <ul class="tree">
        <li>
 <span class="open" onclick="toggle(this)">考勤管理</span>
            <ul class="show">
                <li>日常考勤</li>
                <li>请假申请</li>
                <li>加班/出差</li>
            </ul>
        </li>
        <li><span class="closed" onclick="toggle(this)">信息中心</span>
            <ul class="hide">
                <li>通知公告</li>
                <li>公司新闻</li>
                <li>规章制度</li>
            </ul>
        </li>
        <li><span class="closed" onclick="toggle(this)">协同办公</span>
            <ul class="hide">
                <li>公文流转</li>
                <li>文件中心</li>
                <li>内部邮件</li>
                <li>即时通信</li>
                <li>短信提醒</li>
            </ul>
        </li>
    </ul>
</body>
function toggle(span){//span 指代当前用户点击的 span
    //如果当前 span 的 className 是 open
    if(span.className=="open"){
    //      将当前 span 的 className 改为 closed
        span.className="closed";
    //      将旁边的 ul 的 class 设置为 hide
        //ul.tree ul.show
document.querySelector("ul.tree ul.show").className="hide";
    }else{//否则(如果自己是 closed)
    //      先找到另一个 class 属性为 open 的 openSpan
        //ul.tree span.open
        var openSpan=
            document.querySelector("ul.tree span.open");
    //      如果 openSpan 不为空
        if(openSpan){
    //          将 openSpan 的 className 属性改为 closed
            openSpan.className="closed";
    //      找到 ul 中样式为 show 的 ul，修改 class 为 hide
document.querySelector("ul.tree ul.show").className="hide";
        }
    //      将当前 span 的 className 改为 open
        span.className="open";
    //      找到 ul 中 open 的 span 旁边的 ul
       //ul.tree span.open+ul
document.querySelector("ul.tree span.open+ul").className=
                                                "show";
    }
}
```



Day 02

1. 获取或修改元素的内容：3 个属性：

1. innerHTML：获得/设置元素开始标签和结束标签之间的 html 原文

何时使用：只要获取或修改元素的 HTML 原文时

固定套路：2 个：

1. 批量删除父元素下所有子节点

elem.innerHTML="";

2. 批量替换父元素下所有子节点

elem.innerHTML="所有子元素标签组成的 html 字符串"

技巧：

如果遍历中的操作会减少数组的元素个数，就会影响后续下标

但不影响前边元素的下标——应该从后先前遍历！

option 对象：selected-->可当做 bool 类型用

2. textContent: 获得或设置元素开始标签和结束标签之间的转化后的正文内容。

IE8 不兼容：换为 innerText

1. **元素属性：

核心 DOM：不带 HTML 开头的类型中定义的属性和方法

HTML DOM：在带 HTML 开头的类型中定义的属性和方法

比如：元素：-->HTMLXXXElement 的原型

-->HTMLElement 的原型——所有 HTML 元素的父类型

******以上是 HTML DOM，以下是核心 DOM*****

-->Element 的原型——所有元素的父类型

-->Node 的原型——所有节点的父类型

***一切从属性获取的值和设置到属性上的值必须都是字符串

所以，都要先类型转换，再计算！

获取元素的属性值：4 种：

1. var strValue=elem.getAttribute("属性名");

何时使用：只找一个属性值时

2. elem.attributes 集合：包含了当前元素所有属性的节点对象

elem.attributes[i/"属性名"]-->返回一个属性节点对象

elem.attributes[i/"属性名"].value-->属性值

何时使用：遍历一个元素所有属性时

3. elem.getAttributeNode("属性名")-->返回一个属性节点

elem.getAttributeNode("属性名").value-->返回属性值

修改元素的属性值：2 种：

1. elem.setAttribute("属性名","属性值");

何时使用：只设置一个属性值时

2. elem.setAttributeNode(属性节点对象)

移除元素的属性：2 种：

1. elem.removeAttribute("属性名")

2. elem.removeAttributeNode(属性节点)

判断是否包含指定属性：1 种：

1. elem.hasAttribute("属性名")-->有，true；没有，false

总结：5 个词：

attributes 集合 getAttribute setAttribute removeAttribute hasAttribute

***Property vs Attribute

属性	html 特性
Property:	内存中对象的属性，通常打.访问
Attribute:	指出现在页面 html 代码中的属性

标准属性：Property 等效于 Attribute

比如：a.setAttribute("href","url")——核心 DOM

a.href="url"; ——HTML DOM

今后操作标准属性，用 HTML DOM 的.

因为所有标准属性都被 HTMLXXXElement 及其父类型预定义好了

特例：txt.setAttribute("class","类名");

txt.className="类名";

H5 中自定义属性：只能用 setAttribute,不能直接用.

比如：a.age=18;X——只定义在内存中，无法出现在 html 中

a.setAttribute("age",18);——age 会出现在 html 中

.设置的，不能出现在页面上，就不能被 getAttribute setAttribute 设置的，会出现在页面上，但不能用.访问

所以，h5 规定：凡是自定义属性都要加 data-前缀

获取自定义属性值：a.dataset.属性名

设置自定义属性值：a.dataset.属性名=值

课后练习：*****类数组对象能否转为数组？

数组=Array.prototype.slice.call(类数组对象);

能否对类数组对象排序？2 步：

1. 数组=Array.prototype.slice.call(类数组对象);

2. 数组.sort();

3. 修改样式：

要修改的样式定义在哪儿

1. 仅获得或修改内联样式：每个元素的 style 属性-->style 对象

style 对象中包含了所有 css 样式属性，默认值都是""

elem.style.属性名="值";

问题：无法获得样式表中层叠或继承来的属性

只能访问 elem 的内联样式(style 属性中定义的样式)

2. 获得样式表中层叠或继承来的属性：

DOM 标准：var style 对象=getComputedStyle(elem);

style 对象.属性名

IE8: var style 对象=elem.currentStyle;

style 对象.属性名

问题：仅能获得样式，无法修改样式

解决：要修改：

改内联：只能用 elem.style.属性名="值"

3. 改样式表中的样式：3 步：

1. 获得要修改的样式表对象：

var sheet=document.styleSheets[i];

2. 获得要修改的 CSSRule：

cssRule：样式表中每个大括号包裹的内容

var cssRule=sheet.cssRules[i]

3. 获得或修改 cssRule 中的属性

cssRule.style.属性名=值

问题：牵一发而动全身

解决：都是用直接修改内联样式的属性，来覆盖样式表中的属性

案例

<title>读取并修改元素的内容</title>

```
<style>
    div{float:left; height: 100px; line-height: 100px; }
    #d1,#d3{ background-color: #ccff00; }
    #d2{ cursor: pointer; background-color: #ffcc00; }
    .hide{display:none};
</style>
<script>
    function toggle(d2){
        //找到 d1，同时设置 d1 的 className
        var d1=document.getElementById("d1");
        // 如果 d1 的 className!=""，就改为"",
        //否则改为 hide
        d1.className=d1.className!=""?"":"hide";
    //如果 d2 的 innerHTML 等于"&lt;&lt;",就改为"&gt;&gt;"
     //否则改回 lt
d2.innerHTML=d2.innerHTML=="&lt;&lt;"?"&gt;&gt;": "&lt;&lt;";
    }
</script>
<body>
    <div id="d1">树形列表</div>
    <div id="d2" onclick="toggle(this)">&lt;&lt;</div>
    <div id="d3">内容的主体</div>
</body>
```

<title>读取并修改元素的内容</title>

```
<style>
    select{width:100px;height:85px;}
    div{display:inline-block;width:50px}
</style>
<body>
    <!--假设两个 select 元素，分别保存备选地区列表和选中地区列表
    实现两选择框之间选项的交换：
    包括：当个选中项左右移动
                多个选中项左右移动
                全左移和全右移
要求：两个 select 中的地区都要按照名称首字母排序-->
    <select id="unsel" size="5" multiple>
        <option>Argentina</option>
        <option>Brazil</option>
        <option>Canada</option>
        <option>Chile</option>
        <option>China</option>
        <option>Cuba</option>
        <option>Denmark</option>
        <option>Egypt</option>
        <option>France</option>
        <option>Greece</option>
        <option>Spain</option>
    </select>
    <div>
<button onclick="move(this)">&gt;&gt;</button>
<button onclick="move(this)">&gt;</button>
<button onclick="move(this)">&lt;</button>
<button onclick="move(this)">&lt;&lt;</button>
    </div>
    <select id="sel" size="5" multiple>
    </select>
</body>
window.onload=function(){
    (function(){
        //找到 id 为 unsel 的 select 元素，保存 unsel 中
        var unsel=document.getElementById("unsel");
        unselArr=unsel.innerHTML
//将 unsel 的 innerHTML，替换开始位置的\s*<option>为""
                        .replace(/^\s*<option>/,"")
                    //替换结束位置的<\/option>\s*为""
                        .replace(/<\/option>\s*$/,"")
                    //按<\/option>\s*<option>切割
                        .split(/<\/option>\s*<option>/)
    })();
}
var unselArr=[];
var selArr=[];
function move(btn){
    switch(btn.innerHTML){
        case "&gt;&gt;"://将 unselArr 拼接到 selArr 结尾
            selArr=selArr.concat(unselArr);
            unselArr=[];
            break;
        case "&lt;&lt;"://将 selArr 拼接到 unselArr 结尾
            unselArr=unselArr.concat(selArr);
            selArr=[];
            break;
        case "&gt;"://将 unselArr 中选中的项移动到 selArr 中
            //找到 unsel 中所有 option，保存在变量 opts 中
        var opts=document.querySelectorAll("#unsel>option");
            //i 从 opts.length-1 开始，遍历 opts 中每个 option
            //      到 0 结束，每次递减 1
            for(var i=opts.length-1;i>=0;i--){
            //    如果当前 opt 的 selected 属性有效
                if(opts[i].selected){
    //将 unselArr 中 i 位置的元素删除，用变量 delete 接住，
            // 取 delete 中第 0 个元素，追加到 selArr 中
                    selArr.push(unselArr.splice(i,1)[0]);
                }
```

```
                }
                break;
        }//每次移动完，都要排序
        unselArr.sort();
        selArr.sort();
        show();
}
function show(){ //将每次移动后的两个数组join成html字符串，
                //放到对应 sel 元素的内容中
        //找到 id 为 unsel 的元素，保存在变量 unsel 中
        var unsel=document.getElementById("unsel");
        if(unselArr.length==0){//如果 unselArr 的长度为 0
                unsel.innerHTML="";//就将 unsel 的内容设置为""
        }else{//否则：先拼 unsel 中的内容
                //将拼好的 html 放入,unsel 中
                unsel.innerHTML=
"<option>"+unselArr.join("</option><option>")+"</option>";
        }
        //找到 id 为 sel 的元素，保存在变量 sel 中
        var sel=document.getElementById("sel");
        if(selArr.length==0){//如果 selArr 的长度为 0
                sel.innerHTML="";//就将 sel 的内容设置为""
        }else{//否则：先拼 sel 中的内容
                //     将拼好的 html 放入,sel 中
                sel.innerHTML=
"<option>"+selArr.join("</option><option>")+"</option>";
        }
}
```

<title>修改元素的属性</title>

```
        <!--1. 使用自定义属性实现摇号排名-->
        <ul id="queue">
                <li>Eric</li>
                <li>Scott</li>
                <li>Jerry</li>
                <li>Tom</li>
                <li>Rose</li>
                <li>John</li>
                <li>Smith</li>
                <li>Andy</li>
        </ul>
</body>
window.onload=function(){
        //按 id 找到 ul，保存在变量 ul 中
        var ul=document.getElementById("queue");
        //找到 ul 下所有 li，保存在 lis 中
        var lis=ul.getElementsByTagName("li");
        //将 lis 的 li 个数保存在变量 len 中
        var len=lis.length; //避免动态集合的反复查找
        //定义空数组 nums
        var nums=[];
        //反复生成序号，直到 nums 的元素个数==len
        while(nums.length<len){
        //在 1~len 范围内生成一个随机整数，保存在变量 num 中
                var num=parseInt(Math.random()*len+1);
        //      如果 num 在数组中不存在
                if(nums.indexOf(num)==-1){
                        nums.push(num);//将 num 压入 nums 中
                }
        }
        //遍历 lis 中每个 li，i 从 0 开始，到<len 结束，i 每次增 1
        for(var i=0;i<len;i++){
        //      为当前 li 添加自定义属性 data-i,
        //      值设置为 nums 中相同位置的值
        //        2 种方法：xxx.setAttribute("data-i",?);
        //                  xxx.dataset.i=?;
                lis[i].dataset.i=nums[i];
        }
        //***将类数组对象变为数组类型对象
        lis=Array.prototype.slice.call(lis);
        //数组                  //类数组对象
        lis.sort(function(a,b){//自定义比较器
                return a.dataset.i-b.dataset.i;//比较两 li 的 i 属性
        });
        //遍历 lis 数组中每个 li
        //      将当前元素的 li 变为字符串<li...></li>
        for(var i=0;i<len;i++){
          //字符串                    //li 对象
                lis[i]='<li data-i="'+lis[i].dataset.i+'">'
                        +lis[i].innerHTML
                        +'</li>';
        }//lis 中都变成了 li 元素的 html 字符串
        //将 lis 中的所有 li 字符串无缝拼接，并替换 ul 的内容
        ul.innerHTML=lis.join("");
}
```

<title>实现带样式的表单验证</title>

```
 <body>
<form>
  <h2>增加管理员</h2>
  <table>
    <tr>
        <td>姓名：</td><td>
        <!--当获得焦点时-->
      <input name="name" onfocus="getFocus(this)"onblur=
                                        "valiName(this)"/>
        <!--当失去焦点时-->
          <span>*</span>
        </td>
        <td>
          <div class="vali_Info">
              10 个字符以内的字母、数字和下划线
```

```
            </div>
        </td>
        </tr>
        <tr>
            <td>密码：</td>
            <td>
              <input type="password" name="pwd"onfocus=
                        "getFocus(this)"onblur="valiPwd(this)"/>
             <span>*</span>
            </td>
            <td>
                <div class="vali_Info">6 位数字</div>
            </td>
            </tr>
            <tr>
                <td></td>
                <td colspan="2">
                        <input type="submit" value="保存"/>
                        <input type="reset" value="重填"/>
                </td>
            </tr>
           </table>
    </form>
 </body>
function getFocus(txt){
    txt.setAttribute("class","txt_focus");
    //txt.className="txt_focus";
    //找到旁边 td 中的 div，设置移除 class 属性
    txt.parentNode
        .parentNode
        .querySelector("div")
        .removeAttribute("class");
}
function valiName(txtName){
    txtName.removeAttribute("class");
    //txtName.className="";
    var div=txtName.parentNode
                .parentNode
                .querySelector("div");
    //如果姓名文本框格式正确，则为 div 穿 vali_success
    if(/^\w{1,10}$/.test(txtName.value)){
        div.setAttribute("class","vali_success");
    }else{//否则为 div 穿 vali_fall
        div.setAttribute("class","vali_fail");
    }
}
function valiPwd(txtPwd){
    txtPwd.removeAttribute("class");
    //先查找 txtPwd 旁边的 div，保存在变量 div 中
    var div=txtPwd.parentNode
                .parentNode
                .querySelector("div");
    //如果密码文本框格式正确，则为 div 穿 vali_success
    if(/^\d{6}$/.test(txtPwd.value)){
        div.setAttribute("class","vali_success");
    }else{//否则为 div 穿 vali_fall
        div.setAttribute("class","vali_fail");
    }
}
<title>读取并修改元素的属性</title>
<style>.foreground { z-index: 1; }</style>
<script>
    function show(id){
       /*class*/
        var oldDiv=document.querySelector(".foreground");
        oldDiv.className="";
        var div=document.getElementById(id);
        div.className="foreground";
        /*//将当前处在最上层的 div 清除样式
     var oldDiv=document.querySelector('div[style*="z-index"]');
        //elem.style 对象
        //elem.style.removeProperty 负责移除一个 css 属性
        //elem.style.cssText 获取或设置内联样式字符吕
        //oldDiv.style.removeProperty('z-index')
        oldDiv.style.removeProperty("z-index");
        var div=document.getElementById(id);
        //elem.style.setProperty 设置一个内联属性值
        div.style.zIndex="1";
        //div.style.setProperty("z-index","1");*/
    }
</script>
<h2>实现多标签页效果</h2>
<ul id="tab">
    <li id="tab1" onclick=
         "show('content1')" class="foreground">10 元套餐</li>
    <li id="tab2" onclick="show('content2')">30 元套餐</li>
    <li id="tab3" onclick="show('content3')">50 元包月</li>
    </ul>
    <div id="container">
        <!--<div id="content1" class="foreground">-->
<!--强调：使用 style.设置的属性，默认格式都是:"属性: 值;"-->
        <div id="content1" style="z-index: 1;">
            10 元套餐详情：<br /> 每月套餐内拨打
            100 分钟，超出部分 2 毛/分钟
        </div>
        <div id="content2">
            30 元套餐详情：<br /> 每月套餐内拨打
            300 分钟，超出部分 1.5 毛/分钟
        </div>
        <div id="content3">
```

```
                50 元包月详情：<br /> 每月无限量随心打
            </div>
        </div>
</body>
<title>实现表盘样式的时间显示</title>
    <style>
        #clock{width:100px; height:100px;
                border-radius:50%;
                border:4px solid black;
                position:relative;
        }
        #s{width:2px; height:55px;
                position:absolute; top:5px; left:49px;
                background-color:red;
                transform-origin:50% 45px;
                -webkit-animation:rotateS 60s linear infinite;
        }
        #m{width:4px; height:50px;
                position:absolute; top:10px; left:48px;
                background-color:black;
                transform-origin:50% 40px;
                -webkit-animation:rotateM 3600s linear infinite;
        }
        #h{width:6px; height:45px;
                position:absolute; top:15px; left:47px;
                background-color:black;
                transform-origin:50% 35px;
                -webkit-animation:rotateH 43200s linear infinite;
        }
        @-webkit-keyframes rotateS{
                0%{transform:rotate(0deg)}
                100%{transform:rotate(360deg)}
        }
        @-webkit-keyframes rotateM{
                0%{transform:rotate(0deg)}
                100%{transform:rotate(360deg)}
        }
        @-webkit-keyframes rotateH{
                0%{transform:rotate(0deg)}
                100%{transform:rotate(360deg)}
        }
    </style>
  <body>
    <div id="clock">
        <div id="h"></div>
        <div id="m"></div>
        <div id="s"></div>
        <div id="a1">I</div>
        <div id="a2">II</div>
        <div id="a3">III</div>
        <div id="a4">IIII</div>
        <div id="a5">V</div>
        <div id="a6">VI</div>
        <div id="a7">VII</div>
        <div id="a8">VIII</div>
        <div id="a9">IX</div>
        <div id="a10">X</div>
        <div id="a11">XI</div>
        <div id="a12">XII</div>
    </div>
  </body>
(function(){
        var now=new Date();
        var h=now.getHours();
        h>12&&(h-=12);
        var m=now.getMinutes();
        var s=now.getSeconds();

        var sDeg=s/60*360;
        var mDeg=(m*60+s)/3600*360;
        var hDeg=(h*3600+m*60+s)/(3600*12)*360;

        var sheet=document.styleSheets[1];
        var sRule=sheet.cssRules[4];
        var mRule=sheet.cssRules[5];
        var hRule=sheet.cssRules[6];
        sRule.cssRules[0].style.transform="rotate("+sDeg+"deg)";
sRule.cssRules[1].style.transform="rotate("+(sDeg+360)+"deg)";
        mRule.cssRules[0].style.transform="rotate("+mDeg+"deg)";
mRule.cssRules[1].style.transform="rotate("+(mDeg+360)+"deg)";
})();
```

Day03

1. 创建和删除节点：——核心 DOM
   1. 创建单个节点：3 步：
      1. 创建空节点对象：
         var elem=document.createElement("标签名");
         比如：var a=document.createElement("a");
         <a>
      2. 设置关键属性：(可选)
         比如：a.href="http://tmooc.cn";
         a.innerHTML="go to tmooc"
         go to tmooc
      3. 将新节点挂到指定父元素下：3 种
         追加: parent.appendChild(elem); elem-->lastChild
         插入: parent.insertBefore(elem,oldElem);
         替换: parent.replaceChild(elem,oldElem);

   减少修改 DOM 树（第 3 步）的次数，可减少排版引擎渲染页面的次数，可提高执行效率。
   解决：将所有子元素都添加到父元素下后，再将父元素一次性加入页面。
   2. 添加多个平级子元素：3 步：
      1.先创建一个文档片段：
      文档片段：内存中临时保存多个节点对象的空间
      内存中一个临时的父元素
      var frag=document.createDocumentFragment();
      2.将创建的子元素，先加入到片段中: frag 用法同普通父元素
      frag.appendChild(newElem);
      frag.insertBefore(xxx,xxx);
      frag.replaceChild(xxx,xxx);
      3. 将文档片段作为整体，添加到页面制定父元素下
      片段不出现在 DOM 树中，也不会出现在页面上
3. 删除节点：var 子节点对象=parent.removeChild(子节点对象);
   var 子节点=子节点.parentNode.removeChild(子节点);
2. HTML DOM 常用对象：
   Select 对象：
   属性：sel.options：返回当前 select 下所有 option 对象的集合
   sel.selectedIndex: 获得当前 select 的选中项下标位置
   方法：添加一个 option：sel.add(opt)
   等效于：sel.appendChild(opt);
   移除一个 option：sel.remove(i)
   Option 对象：
   创建 option 对象: var opt=new Option(innerHTML,value);
   等效于: var opt=document.createElement("option");
   opt.innerHTML=?;
   opt.value=?
   属性：opt.selected：返回当前 option 是否被选中，
   可直接当 bool 用
   opt.index：获得 opt 在 select 中的下标位置
1. Select：
   课堂练习：js 如何绑定事件处理函数
   元素的事件处理函数(onxxx),其实就是元素对象的成员：
   元素.onxxx=function(){
   执行事件操作
   this-->当前触发事件的元素对象本身
   }
   当事件发生时，自动执行: 元素.onxxx();
   html 中的 onxxx 属性 vs js 中的 onxxx
   <select onchange="getCities(this)"
   | |
   | |----select---|
   | |
   getCities(sel)-->this:window
2. Table 对象：
   HTMLTableElement：
   方法：每种行分组都有一个 create 方法
   都有一个 delete 方法（除 tbody 没有）
   插入空行: var tr=table.insertRow(i)
   省略 i，表示在末尾追加一行
   删除行: table.deleteRow(i)
   删除行，必须知道要删除行的下标
   属性：tHead tFoot
   *tBodies: 多个 tbody 元素对象的集合
   必须用[i]方式获取每个 tbody
   rows:获得表格中所有 tr 对象
   HTMLTableSectionElement: 表格中每个行分组
   方法：var tr=tbody.insertRow(i)
   //tbody.deleteRow(i);
   注意：i 是相对于 tbody 的序号
   是不包含 thead 中 tr 的序号
   HTMLTableRowElement：表示表格中的每一行
   方法：var td=tr.insertCell(i); ——只能插入 td
   省略 i，表示追加
   tr.deleteCell(i);
   属性：tr.cells: 保存了一行中所有 td 对象
   tr.rowIndex: 保存了当前行的下标，专用于删除行
   HTMLTableCellElement: 表示行中的每一个 td
3. Form 对象:
   如何获得 Form 对象：var form=document.forms[i/name];
   方法: form.submit()，用程序代替自动提交
   事件: onsubmit，当表单提交之前自动触发
   主要用于提交前的验证。
   如何获得 Form 中的数据采集元素：form.elements[i/name];
   方法: elem.focus()

案例

<title>动态创建表格</title>

```
<style>
    table{width:600px; border-collapse:collapse;
        text-align:center;
    }
    td,th{border:1px solid #ccc}
```

```
</style>
<script>
    //假定从服务器端收到 json 字符串
    var json='[{"ename":"Tom", "salary":3500, "age":25},'+
                '{"ename":"John", "salary":3800, "age":28},'+
                '{"ename":"Mary", "salary":3600, "age":25}]';
    //将 json 字符串转为内存中的对象：
    var emps=eval(json);
    window.onload=function(){
        //创建空 table 元素对象
        var table=document.createElement("table");
        //创建空 thead,放入 table 中
        var thead=document.createElement("thead");
        table.appendChild(thead);
        //创建空 tr，放入 thead 中
        var tr=document.createElement("tr");
        thead.appendChild(tr);
        for(var key in emps[0]){//遍历第一个对象中所有属性
            //每遍历一个对象的属性，就创建一个空 th
            var th=document.createElement("th");
            //将当前属性名放入 th 中
            th.innerHTML=key;
            tr.appendChild(th);//将 th 追加到 tr 中
        }
        //创建空 tbody 元素，加入 table 中
        var tbody=document.createElement("tbody");
        table.appendChild(tbody);
    for(var i=0;i<emps.length;i++){//遍历 emps 中每个员工对象
    //  每遍历一个员工就创建一个空 tr,并加入 tbody 中
            var tr=document.createElement("tr");
            tbody.appendChild(tr);
        //    遍历当前员工对象的每个属性
            for(var key in emps[i]){
        // 每遍历一个属性，就创建一个 td 加入 tr 中
                var td=document.createElement("td");
                tr.appendChild(td);
        //          将当前员工的当前属性值放入 td
                td.innerHTML=emps[i][key]
            }
        }//将 table 一次性加入 div 中
    document.getElementById("data").appendChild(table);
    }    </script>
<body>
    <div id="data"></div>
</body>
<title>二级联动列表</title>
<style>    .hide{ display: none; }    </style>
<script>
    /*实现“省”和“市”的级联下拉列表*/
    var cities=[
                [],          /*0 号下标没有元素*/
                [{"name":'东城区',"value":101},
                {"name":'西城区',"value":102},
                 {"name":'海淀区',"value":103},
                 {"name":'朝阳区',"value":104}],
                [{"name":'河东区',"value":201},
                 {"name":'河西区',"value":202},
                 {"name":'南开区',"value":303}],
                [{"name":'石家庄市',"value":301},
                 {"name":'廊坊市',"value":302},
                 {"name":'保定市',"value":303},
                 {"name":'唐山市',"value":304},
                 {"name":'秦皇岛市',"value":304}]
                ];
    function getCities(selProvs){
    //找到 name 为 cities 的 select 元素，保存在变量 selCities
var selCities=document.querySelector('select[name="cities"]');
        //如果 cities[selProvs.selectedIndex].length==0
        if(cities[selProvs.selectedIndex].length==0){
        //    将 selCities 的 class 设置为 hide
            selCities.className="hide";
        }else{//否则(执行下边所有操作)
        //    将 selCities 的 class 设置为""
            selCities.className="";
            selCities.innerHTML="";//清空内容
            selCities.add(new Option("-请选择-",-1));
        //  获取 citis 中相同位置的子数组，保存在 currCities
            var currCities=cities[selProvs.selectedIndex];
        //    遍历 currCities 中的每个城市对象
            for(var i=0;i<currCities.length;i++){
                selCities.add(
        new Option(currCities[i].name ,currCities[i].value)
                );
            }
        }
    }
</script>
<!--当选中项发生改变时触发-->
    <select name="provs" onchange="getCities(this)">
    <option>—请选择—</option><!--当前选中项的下标-->
        <option>北京市</option>   <!--sel.selectedIndex-->
        <option>天津市</option>
        <option>河北省</option>
    </select>
    <select name="cities" class="hide"></select>
<title>动态操作表格</title>
    table{width:500px; border-collapse:collapse;
        text-align:center;
    }
    td,th{border:1px solid #ccc}
```

```
</style>
<script>
    var data=[
    {"id":1001,"name":'可口可乐',"price":2.5,"count":3000},
    {"id":1003,"name":'百事可乐',"price":2.5,"count":5000},
    {"id":1011,"name":'非常可乐',"price":2.3,"count":1000}
    ];
</script>
    <script src="2_1.js"></script>
</head>
<body>    <div id="data"></div>    </body>
window.onload=function(){
    //创建一个 table 对象
    var table=document.createElement("table");
    //在 table 中创建 tHead
    var thead=table.createTHead();
    //在 tHead 中插入一行，tr
    var tr=thead.insertRow();
    //遍历 data 中第一个对象的所有属性
    for(var key in data[0]){
    //      每遍历一个属性就创建一个 th 元素
        var th=document.createElement("th");
    //      设置 th 元素的内容为当前属性名
        th.innerHTML=key;
    //      将 th 加入 tr 中
        tr.appendChild(th);
    }
    //增加操作列
    var th=document.createElement("th");
    th.innerHTML="operate";
    tr.appendChild(th);
    //在 table 下创建一个 tbody
    var tbody=table.createTBody();
    //遍历 data 数组中所有对象
    for(var i=0;i<data.length;i++){
    //    每遍历一个对象，就在 tbody 下插入一个新 tr
        var tr=tbody.insertRow();
    //    遍历当前对象的每个属性
        for(var key in data[i]){
    //        每遍历一个属性，就在 tr 下插入一个新 td
            var td=tr.insertCell();
    //        将当前属性值放入 td 中
            td.innerHTML=data[i][key];
        }//(遍历属性后)给 tr 插入一个空 td——放删除按钮
        var td=tr.insertCell();
        //创建一个删除按钮 button
        var btn=document.createElement("button");
        //将 btn 追加到 td 中
        td.appendChild(btn);
        //修改 button 的内容为"删除"
        btn.innerHTML="删除";
        //为 button 绑定事件处理函数
        btn.onclick=function(){//this->当前点中的删除按钮
            //找到当前按钮所在行
            var tr=this.parentNode.parentNode;
            //获得当前行中第一个格中的 id
            var id=tr.cells[0].innerHTML;
            //提示用户是否删除
            if(confirm("是否删除  "+id+" ?")){
                //点确定：返回 true      否则返回 false
                table.deleteRow(tr.rowIndex);
            }
        }
    }document.getElementById("data").appendChild(table);
}
<title>联动菜单</title>
    <script>
    /*使用 HTML DOM 的方式实现联动菜单*/
    var data=[
        {"id":10,"name":'男装',"children":[
            {"id":101,"name":'正装'},
            {"id":102,"name":'T 恤'},
            {"id":103,"name":'裤衩'}
        ]},
        {"id":20,"name":'女装',"children":[
            {"id":201,"name":'短裙'},
            {"id":202,"name":'连衣裙'},
            {"id":203,"name":'裤子',"children":[
                {"id":2031,"name":'长裤'},
                {"id":2031,"name":'九分裤'},
                {"id":2031,"name":'七分裤'}
            ]},
        ]},
        {"id":30,"name":'童装',"children":[
            {"id":301,"name":'帽子'},
            {"id":302,"name":'套装',"children":[
                {"id":3021,"name":"0-3 岁"},
                {"id":3021,"name":"3-6 岁"},
                {"id":3021,"name":"6-9 岁"},
                {"id":3021,"name":"9-12 岁"}
            ]},
            {"id":303,"name":'手套'}
    ]}
];
<body><div id="category"></div></body>
/*按照给定的数组，创建一个 select 元素*/
function createSelect(arr){
    //创建一个 select 元素
    var sel=document.createElement("select");
    //在 select 中添加一个新 option，内容为"-请选择-",值为-1
```

```
        sel.add(new Option("-请选择-",-1));
        //遍历 arr 中每个对象,形如:{"id":10,"name":'男装'}
        for(var i=0;i<arr.length;i++){
        //      新建一个 option，内容设置为当前对象的 name
        //                      值设置为当前对象的 id
        //                      同时将 opt，加入 select 中
            sel.add(new Option(arr[i].name,arr[i].id));
        }
        //为当前 select 绑定 onchange 事件处理函数
        sel.onchange=function(){//this-->sel
        //从最后一个节点开始，删除当前 select 之后所有节点
            while(this!=this.parentNode.lastChild){
                this.parentNode.removeChild(
                    this.parentNode.lastChild);
            }
            //获得当前 select 选中项的下标
            var i=this.selectedIndex;
        //在创建当前 select 元素的数组(arr)中找对应位置的对象
            var cate=arr[i-1];//cate 获得的就是选中项对应的对象
        if(i!=0&&cate.children){//如果选中对象的 children 属性有效
            //就要创建下级 select:
            //    再次调用 createSelect，传入当前对象的 children
            createSelect(cate.children);
            }
        }
        //找到 id 为 category 的 div，将 select 追加到 div 下
        document.getElementById("category").appendChild(sel);
}
window.onload=function(){
        createSelect(data);
}
<title>表单提交验证</title>
<style>
        .hintText{color: #aaaaaa; font-style: italic; }
        .normalText { color: #000000;}
</style>
<body>
        <h2>发表留言</h2>
        <form name="msgForm">
        姓名：<input name="userName" class="hintText"
                        value="请输入您的姓名"/><br />
        内容：<textarea name="content" class="hintText"
        style="resize:none"     rows="5" cols="30">请输入留言内容
            </textarea><br />
        <label><input type="checkbox"/>
                我已阅读了《常见问题列表》</label><br/>
        <input type="button" value="提交" disabled />
</form> </body>
//当页面加载后
window.onload=function(){
        //找到表单对象
        var form=document.forms[0];
        //获得姓名文本框
        var txtName=form.elements["userName"];
        //绑定姓名文本框的获得焦点和失去焦点事件
        txtName.onfocus=function(){//this-->txtName
            //如果 txtName 的内容是空字符或"请输入您的姓名"
            var reg=/^(\s*|请输入您的姓名)$/;
            if(reg.test(txtName.value)){
                txtName.value="";//清空 txtName 中的内容
            txtName.className="";//移除 txtName 上的 class 属性
            }
        }
        txtName.onblur=function(){
            var reg=/^(\s*|请输入您的姓名)$/;
            if(reg.test(txtName.value)){
                //将 value 重置为默认值
                txtName.value="请输入您的姓名";
                //设置 txtName 的 class 属性为 hintText
                txtName.className="hintText";
            }
        }
        var txtContent=form.elements["content"];
        txtContent.onfocus=function(){
            var reg=/^(\s*|请输入留言内容)$/;
            if(reg.test(txtContent.value)){
                txtContent.value="";
                txtContent.className="";
            }
        }
        txtContent.onblur=function(){
            var reg=/^(\s*|请输入留言内容)$/;
            if(reg.test(txtContent.value)){
                txtContent.value="请输入留言内容";
                txtContent.className="hintText";     }
        }
        //点击已阅读，启用提交
        var chb=form.querySelector('input[type="checkbox"]');
        var btnSubmit= form.querySelector('input[type="button"]');
        chb.onclick=function(){//this-->chb
            btnSubmit.disabled=!chb.checked;
        }
        btnSubmit.onclick=function(){
           if(/^(\s*|请输入您的姓名)$/.test(txtName.value)){
                txtName.focus();//让 txtName 获得焦点
         }else if(/^(\s*|请输入留言内容)$/.test(txtContent.value)){
                txtContent.focus();//让 txtContent 获得焦点
            }else{
                form.submit();
            }    }    }
```

Day04

1. BOM：专门操作浏览器窗口的 API

window：2 个角色：

1. 代替了 ES 中 Global 充当全局对象
2. 封装了浏览器软件/窗口对象的属性和方法——BOM

打开新连接的方式：3 种：

1. 在当前窗口打开，允许后退

HTML：

js：window.open("url","_self");

2. 在当前窗口打开，不允许后退

history：记录*当前窗口**本次*打开后成功访问过的 url

历史记录栈

history.go(-1)：相当于点一次后退按钮

history.go(1)：相当于点一次前进按钮

history.go(0)：相当于点一次刷新按钮

location：当前窗口正在打开的 url 对象（地址栏）

js: location.replace("url");

3. 在新窗口打开，可打开多个

HTML:

js: window.open("url"[,"_blank"])

4. 在新窗口打开，不可打开多个

内存中，窗口都有唯一 name 属性

浏览器规定：同名窗口，只能开一个。后开的会覆盖新开的

其实 target 后的属性值，就是窗口的 name 属性，可自定义。

HTML: <a href="url" target="自定义窗口名">

js: window.open("url","自定义窗口名")

窗口大小和窗口定位：

1. 在打开窗口时，就设定窗口大小：

var opendWindow=window.open("url","name",config)

var config="top=?,left=?,width=?,height=?,

resizable=yes|no,location=no"

2. 大小：window.innerHeight/Width：文档显示区的宽高

文档显示区：仅用于显示网页区域范围大小

window.height/width：整个窗口的宽和高

3. 屏幕大小：

screen.heigth/width：操作系统的桌面分辨率

screen.availHeight/Width：桌面分辨率去掉任务栏的宽度

windows7 下，几乎无差别

4. 调整大小：window.resizeTo(width,height)

window.resizeBy(宽度增量，高度增量);

5. 移动窗口位置：window.moveTo(left,top)

//x y

window.moveBy(x 的增量, y 的增量)

定时器：让程序按指定时间间隔，自动执行***任务***

任务是所有定时器的核心。

2 种：

周期性定时器：让浏览器按照指定时间间隔反复执行同一任务

何时使用：一项任务，连续执行

如何使用：3 步：

1. 定义任务函数：定义了定时器每一次执行的一项任务

function task(){...}

2. 启动定时器：将任务函数放入定时器 API 中，设定时间间隔

最后返回定时器的***序号***

timer=/*window.*/setInterval(task , ms);

全局变量 设置间隔时间

3. 停止定时器：只能通过***序号***,停止定时器

clearInterval(timer);

timer=null;

一次性定时器：让浏览器先等待一定时间后，再自动执行一次任务

如何使用：同周期性定时器，只是将 Interval 换为 Timeout

执行：先等待指定时间，再执行一次，自动退出

定时器的回调函数必须等待主程序执行完才能执行

如果主程序没有执行完，或被阻塞，定时器回调函数永远无法执行。

鄙视题：

```
for(var i=0;i<3;i++){
setTimeout(function(){
console.log(i)
},1);
}//3 个 3
```

固定套路：利用一次性定时器模拟周期性定时器的效果

何时使用：多个任务叠加在一起，避免反复创建多个周期性定时器，任务可能频繁停止和频繁启动

2. navigator: window 中封装浏览器属性和配置信息的对象

cookieEnabled：识别浏览器是否启用 cookie

返回值：true/false

userAgent：保存了浏览器名称和版本的字符串

plugins：封装了所有插件对象的类数组对象

每个 plugin 对象的 name 属性保存了插件的名称

案例

<title>打开新链接方式总结</title>

```
<script>
    /*打开新链接方式总结：
    1. 在当前窗口打开，可后退
    2. 在当前窗口打开，不可后退
    3. 在新窗口打开，可打开多个
    4. 在新窗口打开，不可打开多个*/
    function openUrl1(){
        window.open("http://tmooc.cn","_self");
                                        //target
    }
    function openUrl2(){
        location.replace("http://tmooc.cn");
    }
    function openUrl3(){
    window.open("http://tmooc.cn"/*,"_blank"*/);
                                        //target
```

```
        }
        function openUrl4(){
        window.open("http://tmooc.cn","tmooc");
                                                    //target
        }
    </script>
</head>
<body>
<a href="javascript:openUrl1()">在当前窗口打开，可后退</a><br>
<a href="javascript:openUrl2()">在当前窗口打开，不可后退</a><br>
<a href="javascript:openUrl3()">在新窗口打开，可打开多个</a><br>
<a href="javascript:openUrl4()">在新窗口打开，不可打开多个</a><br> </body>
```

<title>小游戏：点不到的小窗口</title>

```
 <body>   <button onclick="game.start()">开始游戏</button>
 </body>
    var game={
        width:50, //小窗口的宽度
        height:50, //小窗口的高度
        taskH:30, //任务栏高度
        maxTop:null, //最大 top 值
        maxLeft:null, //最大 left 值
        pop:null, //游戏中弹出的小窗口
        start:function(){
        //计算 max 值
        this.maxTop=screen.height-this.height-this.taskH;
        this.maxLeft=screen.width-this.width;
        //0~maxTop 范围内随机生成一个 top 值
        var wTop=parseInt(Math.random()*(this.maxTop+1));
        var wLeft=parseInt(Math.random()*(this.maxLeft+1));
        var config="top="+wTop+",left="+wLeft
                    +",width="+this.width
                    +",height="+this.height
                     +",resizable=yes|no,location=no";
        this.pop=
            /*window.*/open("about:blank","pop",config);
        this.pop.document.write("<img style='width:80px'"
                        +" src='D:\\xiaoxin.gif'"
                        +" onclick='alert(\"约吗？\")'/>"
        );
        //当鼠标进入 pop 窗口时
        this.pop.onmouseover=function(){//this-->pop
            //先获得事件对象：
            var e=window.event||arguments[0];
            //获得鼠标当前相对于显示器的 x,y 坐标
            var x=e.screenX;
            var y=e.screenY;
            //重新反复计算随机位置
            while(true){
        //在 0~maxLeft 之间生成一个随机数，保存在 nextX 中
        var nextX=parseInt(Math.random()*(game.maxLeft+1));
        //在 0~maxTop 之间生成一个随机数，保存在 nextY 中
        var nextY=parseInt(Math.random()*(game.maxTop+1))
        //如果，鼠标位置不在窗口新位置的范围内
        if(!((x>=nextX&&x<=nextX+game.width)&&(y>=nextY&&y<=
nextY+game.height))){
                        //? nextX   nextY    x    y
                        this.moveTo(nextX,nextY);
                        break;//退出循环
                    }
                }
            }
        }
}
```

<title>倒计时</title>

```
 <body>
    <h1>距离放学还有<span id="timer"></span></h1>
    <button onclick="stop(this)">停止倒计时</button>
 </body>
function calc(){//计算当前时间距离目标时间的时间差
    var now=new Date();//获得当前时间对象
    var target=//定义目标时间对象
        new Date(now.toLocaleDateString()+" 17:30:00");
    if(now<=target){
        var ms=target-now;//目标时间-当前时间得到毫秒数
        var s=ms/1000;
    //将毫秒数换算成时，分，秒，每个分量都是 2 位字符
        var h=Math.floor(s/3600);
        h<10&&(h="0"+h);
        var m=Math.floor(s%3600/60);
        m<10&&(m="0"+m);
        s=Math.floor(s%3600%60);
        s<10&&(s="0"+s);
        //拼接结果字符串：时:分:秒
        //找到 span，将内容设置为结果字符串
document.getElementById("timer").innerHTML=h+":"+m+":"+s;
    }else{//如果到达目标时间，就停止定时器
document.getElementById("timer").innerHTML="00:00:00";
        clearInterval(timer);
        timer=null;
    }
}
var timer=null;
window.onload=function(){
    calc();
    timer=setInterval(calc,1000);
}
```

```
function stop(btn){
    if(timer){
        clearInterval(timer);
        timer=null;
        btn.innerHTML="启动倒计时";
    }else{
        calc();
        timer=setInterval(calc,1000);
        btn.innerHTML="停止倒计时";
    }   }
```

<title>Js 版带表盘的时钟</title>

```
 <body>
  <div id="clock">
    <div id="h"></div> <div id="m"></div>
    <div id="s"></div>
    <div id="a1">I</div>
    <div id="a2">II</div>
    <div id="a3">III</div>
    <div id="a4">IIII</div>
    <div id="a5">V</div>
    <div id="a6">VI</div>
    <div id="a7">VII</div>
    <div id="a8">VIII</div>
    <div id="a9">IX</div>
    <div id="a10">X</div>
    <div id="a11">XI</div>
    <div id="a12">XII</div>
  </div>
 </body>
window.onload=function(){
    setInterval(function(){//匿名函数回调
        var now=new Date();
        var divH=document.getElementById("h");
        var divM=document.getElementById("m");
        var divS=document.getElementById("s");
        var h=now.getHours();
        h>12&&(h-=12);
        var m=now.getMinutes();
        var s=now.getSeconds();
        var sDeg=s/60*360;
        var mDeg=(m*60+s)/3600*360;
        var hDeg=(h*3600+m*60+s)/(3600*12)*360;
        divH.style.transform="rotate("+hDeg+"deg)";
          //内联样式
        divM.style.transform="rotate("+mDeg+"deg)";
        divS.style.transform="rotate("+sDeg+"deg)";
    },1000);
}
```

<title>窗口右下角消息弹出框</title>

```
 <body>   <div id="msg"> <a>X</a>   </div> </body>
var adv={
    duration:1000,//总时长 3 秒
    interval:30, //每步间隔 50 毫秒
    div:null, //保存移动的 div 对象
    height:0, //div 的总高度
    init:function(){
        //找到 div 对象，保存在 div 中
        this.div=document.getElementById("msg");
        /*获得 div 对象高度，保存在 height 中*/
        var style=getComputedStyle(this.div);
                //获得 div 对象最终应用的所有样式属性
        this.height=parseFloat(style.height);
    },
    moveUpStep:function(){//向上移动一步
        var self=this;//用局部变量 self 转接 this 指的当前对象
        //获得 div 现在最终应用的 bottom 值
        var style=getComputedStyle(self.div);
        var bottom=parseFloat(style.bottom);
        //将 bottom 值+=height*interval/duration;
        bottom+=self.height*self.interval/self.duration;
    //将 bottom 的值设置回 div 的内联样式中，替代外部样式
        self.div.style.bottom=bottom+"px";
        //如果 bottom 值<=0，就再次注册一次性定时器
        if(bottom<=0){//必须使用 adv 对象调用 moveUpStep
setTimeout(function(){self.moveUpStep()},self.interval);
        }
    },
    startMoveUp:function(){//让 div 开始向上移动
        var self=this;
        //注册一次性定时器，放入 moveUpStep
setTimeout(function(){self.moveUpStep()},self.interval);
    },
    moveDownStep:function(){//向下移动一步
        var self=this;
        //获得 div 现在最终应用的 bottom 值
        var style=getComputedStyle(self.div);
        var bottom=parseFloat(style.bottom);
        //将 bottom 值-=height*interval/duration;
        bottom-=self.height*self.interval/self.duration;
    //将 bottom 的值设置回 div 的内联样式中，替代外部样式
        self.div.style.bottom=bottom+"px";
        //如果 bottom 值>=-height，
        if(bottom>=-self.height){
        //    就再次注册一次性定时器
            setTimeout(function(){
                self.moveDownStep();
            },self.interval);
        }else{//否则
        //  在注册一次 startMoveUp，设置间隔为 5000 毫秒
            setTimeout(function(){
```

```
                self.moveUpStep();
            },5000);
        }
    },
    startMoveDown:function(){
        var self=this;
        //注册一次性定时器
        setTimeout(function(){
            self.moveDownStep();
        },self.interval);
    }
}
window.onload=function(){
    adv.init();
    adv.startMoveUp();
   document.querySelector("#msg>a").onclick=function(){
        adv.startMoveDown(); }
}
```

<title>倒计时自动关闭/跳转页面</title>

```
<script>
    function calc(){
        //找到 id 为 msg 的 span
        var span=document.getElementById("msg");
        //将 span 中的数字部分，转为整数，保存在 n 中
        var n=parseInt(span.innerHTML);
        //将 n 先递减 1，再比较
        if(--n>0){//如果 n>0
        //将 n 拼接上 span 内容中剩余部分，放回 span 中
            span.innerHTML=n+"秒钟后自动关闭";
            timer=setTimeout(calc,1000);
        }else{ /*window.*/close();}//关闭当前窗口
    }
    var timer=null;
    window.onload=function(){
        timer=setTimeout(calc,1000);
    } </script>
<body>
    <span id="msg">5 秒钟后自动关闭</span><br>
    <a href="javascript:clearTimeout(timer)">留在本页</a>
</body>
```

<title>navigator 对象常用属性</title>

```
<script>
    if(navigator.cookieEnabled){
    document.write("已启用 cookie，请妥善保存个人信息");
    }else{
    document.write("cookie 已禁用，自动登录功能无法使用");
    }
    document.write("<br>");
    document.write(navigator.userAgent);
    document.write("<br>");
    var ua=navigator.userAgent;
    var browser="unknown";
    if(ua.indexOf("MSIE")!=-1){
        browser="MSIE";
    }else if(ua.indexOf("Firefox")!=-1){
        browser="Firefox";
    }else if(ua.indexOf("OPR")!=-1){
        browser="OPERA";
    }else if(ua.indexOf("Chrome")!=-1){
        browser="Chrome";
    }else if(ua.indexOf("Safari")!=-1){
        browser="Safari";
    }else if(ua.indexOf("Trident")!=-1){
        browser="IE 11";
    }
    var ver="";
    if(browser=="unknow"){
        alert("不支持");
    }else if(browser=="OPERA"){
        var i=ua.indexOf("OPR");
        i+=4;
        ver=parseFloat(ua.slice(i,i+4));
    }else if(browser=="IE 11"){
        ver="11";
    }else{
        var i=ua.indexOf(browser);
        i+=browser.length+1;
        ver=parseFloat(ua.slice(i,i+4));
    }
    alert(browser+" "+ver);
    var ps=navigator.plugins;
    for(var i=0;i<ps.length;i++){
        if(ps[i].name=="QQMusic"){
            alert("支持 QQMusic");
            break;
        }
    }
</script>
<body>
```

```
> var n=0;
  function task(){
      if(n<5){
          console.log("该起床了");
          timer=setTimeout(task,1000);
          n++;
      }
  }
  var timer=null;
  timer=setTimeout(task,1000);
< 1
5 该起床了
>
```

Day05

1. BOM 常用对象：

window navigator history location document screen event

navigator: 保存浏览器配置信息的对象

cookieEnabled: 判断是否启用 cookie

userAgent: 判断浏览器的名称和版本号

plugins: 保存浏览器中所有插件信息的集合

screen: 保存显示屏信息的对象

history: 当前窗口的历史记录栈

history.go(-1): 后退

history.go(0): 刷新

history.go(1): 前进

location: 指代当前窗口正在访问的 url 地址对象

location.href: 保存了当前窗口正在访问的 url 地址

设置 href 属性为新 url，会在当前窗口打开新 url

location.assign(url): 设置当前窗口的新 url

location.reload(true/false): true: 无论是否更改，都获取最新

false: 被修改的页面，重新获取

未被修改的页面，从缓冲中取

2. ***事件：浏览器自动触发的或用户手动触发的对象状态的改变

DOM Level2 Event 标准：

IE8：自成体系！

事件处理函数：当事件发生时，自动执行的函数对象

其实就是：on 事件名

如何绑定事件处理函数：3 种：

1. 在 html 元素的开始标签中设置事件处理函数属性：

比如：<button onclick="fun(this)"></button>

| |

|-|----------button

| |

fun(btn){...this-->window; btn-->button...}

执行：fun();

2. 在 js 中动态绑定事件处理函数：elem.on 事件名=函数对象

比如：btn.onclick=function(){...this-->btn...}

执行：btn.onclick();

问题 1：同一元素的同一事件处理函数，只能绑定一个函数对象

问题 2：无法修改事件触发的顺序。

3. 在 js 中动态绑定事件处理函数，

可同时绑定多个，可修改事件触发顺序

DOM 标准：elem.addEventListener(

"事件名",函数对象,是否在捕获阶段触发)

IE8: elem.attachEvent("on 事件名",函数对象);

比如：btn.addEventListener("click",fun,false/true);

btn.attachEvent("onclick",fun);

执行：btn.onclick();

所以，fun(){this-->btn}

***事件周期：从事件触发到各级事件执行完的全过程

DOM 标准：3 个阶段：(从根开始，由外向内，到目标元素)捕获

(实际触发事件的对象)目标触发

(从目标向外，到根结束)冒泡触发

IE8：2 个阶段：(实际触发事件的对象)目标触发

(从目标向外，到根结束)冒泡触发

是否可修改事件执行顺序:可以

修改 addEventListener 的第三个参数为 true，可在捕获阶段提前触发

同一事件不可触发 2 次！

事件对象：当事件发生时，自动创建的封装事件信息的对象

获得事件对象：

1. 在 html 中绑定事件处理函数：在 html 中显式使用 event

<button onclick="fun(this,event)"></button>

| | |

|----------button--event

| | |

fun(btn,e){...e-->获得事件对象...}

兼容性最好的。

2. 在 js 中动态绑定事件处理函数：

DOM 标准：事件对象作为事件处理函数第一个参数，传入函数中

arguments[0]-->事件对象

IE8：事件对象是 window 下的一个全局属性:window.event

兼容：var e=window.event||arguments[0];

IE8 DOM

固定套路：

1. 取消冒泡：在当前事件处理函数执行结尾，阻止继续向上冒泡

DOM：e.stopPropagation();

IE: e.cancelBubble=true;

兼容代码：if(e.stopPropagation){

e.stopPropagation();

}else{

e.cancelBubble=true;

}

2. 利用冒泡：

***优化：多个子元素，定义了相同的事件处理函数，其实只需要将事件处理函数在父元素上定义一次即可。

Why: 每个处理函数的绑定都是一个对象

处理函数绑定的越多，网页的执行效率越低。

减少处理函数绑定的次数，可提高执行效率。

获得目标元素：var target=e.srcElement||e.target

IE DOM

目标元素 vs this:

this:会随事件冒泡而改变

目标元素：不受冒泡影响，始终保存最初的目标元素对象

3. 取消事件：事件执行过程中，发生异常状况，可阻止事件触发

比如：onsubmit：自动提前之前触发

如果验证未通过，可取消继续提交

何时使用：取消元素默认的事件行为。

1. 在 html 中绑定事件处理函数：2 个 return

比如：<form onsubmit="return valiAll()">

valiAll(){... ...return true/false}

如果返回 true，继续执行

如果返回 false，取消执行

2. 在 js 中动态绑定处理函数：（也可用与 html 绑定中）

如何取消：if(e.preventDefault){

e.preventDefault();//DOM

}else{

e.returnValue=false;//IE

}

1. ***事件：

事件坐标：3 种：

1. 相对于显示器左上角：

最大值 screen.availWidth/Height

鼠标位置：e.screenX/Y

2. 相对于文档显示区左上角：

最大值：window.innerWidth/Height

鼠标位置：e.clientX/Y

e.x/y

3. 相对于父元素左上角：

最大值：父元素的宽和高

鼠标位置：e.offsetX/offsetY

2. 页面滚动：

事件：window.onscroll

滚动的 top 距离：document.documentElement.scrollTop

||document.body.scrollTop

3. cookie: 客户端保存用户个人信息的文件

每个 cookie 是一个 key/value 对儿

key 表示存储的属性名

value 表示属性的值

何时使用：只要在客户端持久缓存数据

什么样的数据放在客户端：用户的个人信息，操作，配置

为什么要持久保存：即使当前会话结束，其他网页或下次启动会话

还需要使用该数据

cookie 属性：

expires: 过期时间：晚于当前时间的时间点

下次访问相同网页时，浏览器会检查 cookie

如果过期，才会删除 cookie

domain: 规定了 cookie 只能被哪个域名使用

可设置

如何使用 cookie：

1. 保存：document.cookie="key=value"
2. 读取：document.cookie

var cookies=document.cookie.split("; ");

cookies[i]: "key=value"

案例

<title>事件处理</title>

```
<script>
    /*1. 体验事件冒泡*/
    function fun(){//js 动态绑定，this-->元素对象
        var e=window.event||arguments[0];
        this.style.backgroundColor="yellow";
        var target=e.srcElement||e.target;
        alert(target.className);
        this.style.backgroundColor="";
        //取消冒泡
        /*if(e.stopPropagation){
            e.stopPropagation();
        }else{
            e.cancelBubble=true;
        }*/
    }
    window.onload=function(){
        var divs=document.querySelectorAll("div");
        divs[0].addEventListener(
"click",fun,false);
        divs[1].addEventListener(
"click",fun,false);
        divs[2].addEventListener(
"click",fun,false);
    }  </script>
<body>
    <div class="d1"><!--onclick="fun()"-->
        <div class="d2"><!--onclick="fun()"-->
        <div class="d3"><!--onclick="fun()"--></div>
        </div>
    </div>
</body>
```

<title>取消与利用冒泡</title>

```
<meta charset="utf-8"/>
<link rel="stylesheet" href="css/3.css"/>
<script>
    /*2. 利用冒泡，简化计算器程序*/
    function calc(e){
    //从 e 中获得目标元素按钮,保存在 target 中
        var target=e.srcElement||e.target;
    if(target.nodeName=="BUTTON"){//只有点按钮
    //按 id 找到文本域对象，保存在 sc 中
    var sc=document.getElementById("sc");
            //如果 target 的内容是"C"
            if(target.innerHTML=="C"){
            sc.innerHTML="";//清空 sc 的内容
            }else if(target.innerHTML=="="){
            //否则 如果 target 的内容是"="
```

```
            // 将 sc 的内容传入 eval 中执行，结果再放回 sc
                    try{sc.innerHTML=eval(sc.innerHTML);
                            }catch(err){
                                sc.innerHTML=err.message;
                            }
                        }else{//否则
                    //将 target 的内容，追加到 sc 的内容上
    sc.innerHTML+=target.innerHTML;
                        }
                    }
                }
            </script>
    <body>
        <div onclick="calc(event)"><!--this-div-->
            <button>1</button>
            <button>2</button>
            <button>3</button>
            <button>4</button><br>
            <button>C</button>
            <button>+</button>
            <button>-</button>
            <button>=</button>
        </div>
        <textarea id="sc" style="resize:none;width:200px;
height:50px;" readonly></textarea>
    </body>
```

<title>阻止自动表单提交</title>

```
 <body>
    <!--1. 使用 HTML 绑定事件处理函数时
    <form onsubmit="return valiAll()">-->
    <form onsubmit="valiAll(event)">
        <h2>增加管理员</h2>
        <table>
            <tr>
            <td>姓名：</td><td>
<input name=
    "name" onfocus="getFocus(this)" onblur="valiName(this)"/>
    <span>*</span>
    </td>
    <td>
    <div class="vali_Info">10 个字符以内的字母、数字和下划
线的组合
    </div>
     </td>
    </tr>
    <tr>
    <td>密码：</td>
    <td>
<input type="password" name="pwd" onfocus="getFocus(this)"
onblur="valiPwd(this)"/>
    <span>*</span>
    </td>
    <td>
        <div class="vali_Info">6 位数字</div>
    </td>
        </tr>
        <tr>
            <td></td>
            <td colspan="2">
<!--取消事件，2 个 return-->
<input type="submit" value="保存"/>
<input type="reset" value="重填"/>
            </td>
        </tr>
    </table>
    </form>
function getFocus(txt){
    txt.setAttribute("class","txt_focus");
    //txt.className="txt_focus";
    //找到旁边 td 中的 div，设置移除 class 属性
    txt.parentNode
        .parentNode
        .querySelector("div")
        .removeAttribute("class");
}
function valiName(txtName){
    txtName.removeAttribute("class");
    //txtName.className="";
var div=txtName.parentNode.parentNode.querySelector("div");
    //如果姓名文本框格式正确，则为 div 穿 vali_success
    if(/^\w{1,10}$/.test(txtName.value)){
        div.setAttribute("class","vali_success");
        return true;
    }else{//否则为 div 穿 vali_fall
        div.setAttribute("class","vali_fail");
        return false;
    }
}
function valiPwd(txtPwd){
    txtPwd.removeAttribute("class");
    //先查找 txtPwd 旁边的 div，保存在变量 div 中
    var div=txtPwd.parentNode
            .parentNode
            .querySelector("div");
    //如果密码文本框格式正确，则为 div 穿 vali_success
    if(/^\d{6}$/.test(txtPwd.value)){
        div.setAttribute("class","vali_success");
        return true;
    }else{//否则为 div 穿 vali_fall
        div.setAttribute("class","vali_fail");
```

```
            return false;
        }
}
function valiAll(e){
var rname=valiName(document.forms[0].elements["name"]);
        var rpwd=valiPwd(document.forms[0].elements["pwd"]);
        if(!(rname&&rpwd)){
            if(e.preventDefault){
                e.preventDefault();//DOM
            }else{
                e.returnValue=false;//IE
            }
        }
}
```

<title>在当前显示区范围内实现点不到的小方块</title>

```
        <style>
            div{position:fixed;width:50px;height:50px;
                background-color:pink;
            }
        </style>
        <script src="js/6.js"></script>
  </head>
  <body>
        <div id="pop" onclick="alert('约吗')"></div>
  </body>
var game={ div:null, //要移动的小 div 对象
        maxLeft:0,maxTop:0,//小 div 最大可用 left 值和 top 值
        WIDTH:0, HEIGHT:0,//小 div 的宽和高，固定
        init:function(){ var self=this; //留住 this
            self.div=document.getElementById("pop");
            var style=null;
            if(getComputedStyle){
                style=getComputedStyle(self.div);
            }else{
                style=self.div.currentStyle;
            }
            self.WIDTH=parseFloat(style.width);
            self.HEIGHT=parseFloat(style.height);
            self.maxLeft=window.innerWidth-self.WIDTH;
            self.maxTop=window.innerHeight-self.HEIGHT-30;
        },
        start:function(){//启动游戏
            var self=this;
            //随机生成新的 left 和 top，保存在 l 和 t 中
            var l=Math.floor(Math.random()*(self.maxLeft+1));
            var t=Math.floor(Math.random()*(self.maxTop+1));
            //修改 div 的 left 和 top 分别为 l 和 t
            self.div.style.left=l+"px";
            self.div.style.top=t+"px";
            self.div.onmouseover=function(){//this-->div
                while(true){
                //随机生成新的 left 和 top，保存在 l 和 t 中
var l=Math.floor(Math.random()*(self.maxLeft+1));
var t=Math.floor(Math.random()*(self.maxTop+1));
                    //获得事件对象 e
                    var e=window.event||arguments[0];
                    //获得鼠标相对于显示区的位置 x，y
                    var x=e.clientX||e.x;
                    var y=e.clientY||e.y;
                    console.log(x+","+y);
                //如果 div 的新位置 l 和 t，不包含鼠标位置 x 和 y
                    if(!(x>=l&&x<=l+self.WIDTH
                        &&y>=t&&y<=t+self.HEIGHT)){
                    //      就设置 div 的 left 为 l，top 为 t
                        self.div.style.left=l+"px";
                        self.div.style.top=t+"px";
                        break;//      退出循环
                    }
                }
            }
        }
}
window.onload=function(){
        game.init();       game.start();
}
```

<title>在某一特定 div 范围内实现点不到的小方块</title>

```
  <body>
        <div id="outer">
            <div id="pop" onclick="alert('约吗？')"></div>
        </div>
  </body>
var game={
        maxTop:0, maxLeft:0,
        height:0, width:0,
        div:null,
        init:function(){var self=this;//留住 this
            //找到 id 为 pop 的 div 保存在当前对象的 div 中
            self.div=document.getElementById("pop");
            //获得当前对象的 div 计算后的样式，存在 style 中
            var style=null;
            if(getComputedStyle){
                style=getComputedStyle(self.div);
            }else{
                style=self.div.currentStyle;
            }
//获得 style 中 height 转为浮点数，存在当前对象的 height 中
            self.height=parseFloat(style.height);
//获得 style 中 width 转为浮点数，存在当前对象的 width 中
            self.width=parseFloat(style.width);
            //找到 id 为 outer 的 div，保存在 outer 中
```

```
        var outer=document.getElementById("outer");
        //获得 outer 对象计算后的样式，存在 style 中
        if(getComputedStyle){style=getComputedStyle(outer);
        }else{style=outer.currentStyle;}
    //当前对象的 maxTop=style 的 height-当前对象的 height;
        self.maxTop=parseFloat(style.height)-self.height;
    //当前对象的 maxLeft=style 的 width-当前对象的 width;
        self.maxLeft=parseFloat(style.width)-self.width;
        self.div.onmouseover=function(){
            while(true){
            //在 0~当前对象的 maxLeft 之间随机生成 l
        var l=Math.floor(Math.random()*(self.maxLeft+1));
            //在 0~当前对象的 maxTop 之间随机生成 t
        var t=Math.floor(Math.random()*(self.maxTop+1));
                //获得事件对象
                var e=window.event||arguments[0];
                //获得鼠标相对于外层元素的位置 x,y
                //                (e.offsetX/offsetY)
                var x=e.offsetX, y=e.offsetY;
                //如果 l 和 t 的新位置不包含 x，y
                if(!(x>=l&&x<=l+self.width
                    &&y>=t&&y<=t+selt.height)){
                //    设置当前对象的 div 的 left 为 l
                    self.div.style.left=l+"px";
                //    设置当前对象的 div 的 top 为 t
                    self.div.style.top=t+"px";
                //    退出循环
                    break;
                }    }    }    }    }
window.onload=function(){
    game.init();
    console.log(game);
}
```

<title>根据页面滚动位置显示浮动框</title>

```
    <script>
        window.onscroll=function(){
            //获得已滚动的 top 距离，保存在 top 中
            var top=document.documentElement.scrollTop
                ||document.body.scrollTop;
            //找到 id 为 toTop 的 div，保存在 toTop 中
            var toTop=document.getElementById("toTop");
            if(top>=300){//判断 top>=300
                //将 toTop 显示出来
                toTop.style.display="block";
            }else{//否则
                toTop.style.display="none";//将 toTop 隐藏
            }
        }</script>
<div id="toTop"><a href="#">返回顶部</a> </div>
```

<title>使用 Cookie</title>

```
    <h2>用户登录</h2>
    <form action="8_loginResult.html" method="post">
    用户名：<input name="userName" /><br/>
    密码：<input name="userPwd" type="password" /><br/>
<input type="radio" name="noLogin" value="7"/>一周内不在登录
<input type="radio" name="noLogin" value="14"/>两周内不在登录
    <input type="radio" name="noLogin" value="30"/>一月内不在登录<br/>
<input type="button" onclick="submitForm()" value="登录"/>
    </form>
</body>
window.onload=function(){
    var cookies=document.cookie.split("; ");
    for(var i=0;i<cookies.length;i++){
        if(cookies[i].indexOf("userName")!=-1){
            location.replace("8_loginResult.html");
        }    }    }
function submitForm(){
    var form=document.forms[0];
    var userName=form.userName.value;
    var userPwd=form.userPwd.value;
    //获得所有单选框
    var rads=form.noLogin;
    for(var i=0,len=rads.length;i<len;i++){
        if(rads[i].checked){
            var expires=new Date(new Date().getTime()
                    +rads[i].value*24*3600*1000);
            document.cookie="userName="+userName
                    +";expires="+expires.toGMTString();
            document.cookie="userPwd="+userPwd
                    +";expires="+expires.toGMTString();
            break;
        }
    }form.submit();
}
<body>
    <h2>欢迎回来：<span id="uname"></span></h2>
</body>
window.onload=function(){
    var cookies=document.cookie.split("; ");
    for(var i=0;i<cookies.length;i++){
        if(cookies[i].indexOf("userName")!=-1){
            var eqi=cookies[i].indexOf("=");
            var userName=cookies[i].slice(eqi+1);
            document.getElementById("uname")
                    .innerHTML=userName;
            break;    }
    }    }
```