JavaScript进阶

操作CSS样式

在JS中操作CSS属性命名上的区别

CSS中写法	JS中的写法	说明
color	**color**	如果样式名只有一个单词，写法相同
font-size	fontSize	如果有多个单词，去掉单词之间短横，并且使用驼峰命名法

通过js代码可以在网页运行过程中动态修改某个元素的样式，有如下两种方式：

方式一：

元素.style.样式名 = 样式值
每条语句只能修改一个样式

方式二：

元素.className = "类名"
先创建好一个类样式，直接给这个元素设置一个类的属性，一条语句可以修改多个样式。

示例：点按钮，修改p标签的字体、颜色、大小

效果

代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>使用js改变css样式</title>
    <style type="text/css">
        .one{
            font-size: 20px;
            color: pink;
            font-family: 仿宋;
        }
    </style>
</head>
<body>
<script type="text/javascript">
    //改变几个样式
    function changeCss() {
        //得到p1这个元素
        var p1 = document.getElementById("p1");
        //修改p1的字体大小
        p1.style.fontSize="40px";
        //修改p1的颜色
        p1.style.color="blue";
        //修改p1的字体
        p1.style.fontFamily="黑体";
    }
    //修改类样式
    function changeClass() {
        //得到p2这个元素
        var p2 = document.getElementById("p2");
        //设置类名
        p2.className="one";
    }
</script>
<p id="p1">我是第一段</p>

<p id="p2">我是第二段</p>
```

```
<input type="button" value="改变几个样式" onclick="changeCss()">
<input type="button" value="改变几个类样式" onclick="changeClass()">
</body>
</html>
```

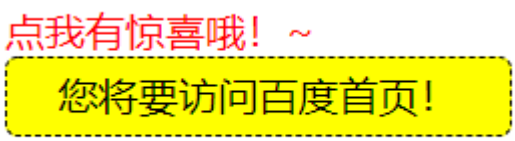
网页的变化

```
<body> == $0
    <p id="p1" style="font-size: 40px; color: blue; font-family: 楷体;">
        我是第一段
    </p>
    <p id="p2" class="two">
        我是第二段
    </p>
    <hr>
    <input type="button" value="改变几个样式" onclick="changeCss()">
    <input type="button" value="改变类样式" onclick="changeClass()">
```

案例：鼠标移动到文字上显示提示文字

display样式	说明
none	设置这个元素不可见
block	设置为块级元素，可见。换行
inline	设置为内联元素，可见。不换行

案例效果：



实现步骤：

1. 网页上有一个a标签链接，a标签下有一个不可见的div。
2. 鼠标移到a标签，设置div的样式，让div可见
3. 鼠标移出a标签，设置div的样式，让div不可见。

案例代码：

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>鼠标移动显示文字</title>
    <style type="text/css">
        a:link{
            /*去下划线*/
            text-decoration: none;
        }
        a:hover{
            color: pink;
        }
        #xf{
            display: none;
            border:1px dashed black;
            width: 200px;
            height: 30px;
            text-align: center;
            line-height: 30px;/*线与线之前的距离*/
            border-radius: 5px;/*向元素添加圆角*/
            background-color: yellow;
        }
    </style>
    <script type="text/javascript">
        window.onload=function () {
            //设置鼠标移上事件
            document.getElementById("xd").onmouseover=function () {
                //改变xf的属性，让他显示
                document.getElementById("xf").style.display="block";
            };
            //设置鼠标移出事件
            document.getElementById("xd").onmouseout=function () {
                //让xf隐藏
                document.getElementById("xf").style.display="none";
            }
        }
    </script>
</head>
<body>
<a href="#" id="xd">点我有惊喜哟！</a>
<div id="xf">您将要访问百度首页！</div>
</body>
</html>
```

正则对象

作用： 可以用来匹配或查找某个字符，如果字符串与指定的表达式匹配，返回true，如果不匹配返回false。

正则表达式不仅用在JS中，大部分高级语言都支持正则表达式，而且语法相同。

正则表达式规则：

符号	作用
[a-z]	[] 表示一个字符，- 表示一个区间。a到z之间的任何一个字符
[xyz]	x或y或z，三个中间的一个字符
[\^xyz]	在[]里面\^，表示取反，除了xyz之外的字符
\d	代表数字
\w	代表单词，包含以下字符：a-zA-Z0-9_
.	代表任意字符，如果要使用匹配点号，要转义\.
()	()代表一个分组
{n}	前面的字符出现n次
{n,}	前面的字符出现大于等于n次
{n,m}	前面的字符出现n到m之间的次数，包头又包尾
+	前面的字符出现1~n次
*****	前面的字符出现0~n次
?	前面的字符出现0~1次
|	或
^	出现在开头，表示必须匹配这个头
$	出现结尾，表示必须匹配这个结尾。

正则表达式举例

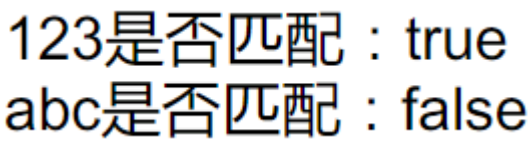
正则表达式	匹配字符串
\d{3}	包含3个数字即可：a123b
\^\d{3}	以3个数字开头：123b
\d{3}$	以3个数字结尾：a123
\^\d{3}$	必须是3个数字：123
[a-d]	小写的a到d中的一个字符，中括号表示匹配1个字符
[xyz]	x或y或z
ab{2}	a后面出现2次b：abb
ab{2,}	a后面出现2次及以上的b：abb或abbb或abbbb
ab{3,5}	a后面出现3~5次b：abbb或abbbb或abbbbb
ab+	a后面出现1~n次b：ab或abb或abbb
ab*	a后面出现0~n次b：a或ab或 abbb
ab?	a后面出现0~1次b：a或ab
hi|hello	字符串里有hi或者hello
(b|cd)ef	表示bef或cdef
\^.{3}$	表示有任意三个字符的字符串
[\^a-zA-Z]	中括号内部的\^，表示不出现，即不出现：大小写字母

创建正则表达式的方式

方式1：

```
var 变量名 = new RegExp("正则表达式")
```

Regular Expression 正则表达式



```
<script type="text/javascript">
    //创建一个匹配3个数字的正则表达式
    var reg = new RegExp("^\\d{3}$");
    //test(字符串)，如果匹配返回true，否则返回false
    document.write("123是否匹配：" + reg.test("123") + "<br/>");
    document.write("abc是否匹配：" + reg.test("abc") + "<br/>");
</script>
```

方式2：

```
var 变量名 = /正则表达式/
```

```
var reg = /^\d{3}$/;
//test(字符串)，如果匹配返回true，否则返回false
document.write("123是否匹配：" + reg.test("123") + "<br/>");
document.write("abc是否匹配：" + reg.test("abc") + "<br/>");
```

两种方式的区别：

1. 正则表达式是写在一个字符串中，\符号需要转义。字符串可以定义成一个变量，灵活一些。
2. 固定的写法，就是正则表达式，\不需要转义。

匹配模式：

可以给正则表达式在匹配的时候指定其它一些选项 i 忽略大小写比较

More Actions匹配模式的两种写法

- 一、var 变量名 = new RegExp("正则表达式", "匹配模式");
- 二、var 变量名 = /正则表达式/匹配模式;

```
<script type="text/javascript">
    //方式一：
    //var reg = new RegExp("abc","i");  //正则表达式
    //方式二：
    var reg = /abc/i;  //正则表达式

    document.write(reg.test("abc") + "<br/>");
    document.write(reg.test("ABC") + "<br/>");
    document.write(reg.test("aBc") + "<br/>");
</script>
```

正则表达式中常用的方法

JS中正则表达式的方法	说明
boolean test("字符串")	作用： 判断正则表达式是否与参数字符串匹配 参数：需要匹配的字符串 返回：匹配返回true，否则返回false

案例：校验表单

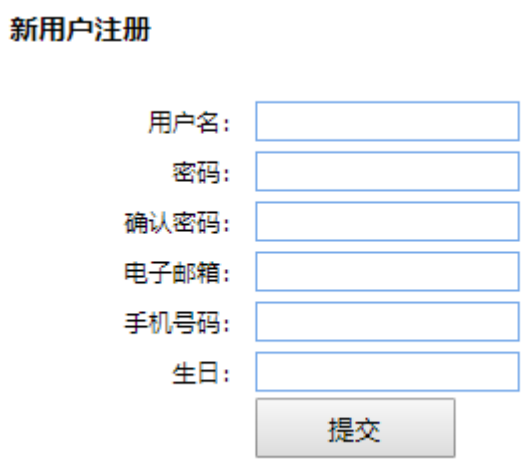
onsubmit事件的说明：如果这个事件中的函数返回true，表单可以提交，否则不能提交

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script type="text/javascript">
        //如果这个函数返回True，表单可以提交，否则不能提交
        function checkAll() {
            return true;
        }
    </script>
</head>
<body>
<h1>用户注册</h1>
<form action="server" method="get" onsubmit="return checkAll()">
    用户名:
    <input type="text" name="user"><br>
    <input type="submit" value="注册">
</form>
</body>
</html>
```

案例需求：

用户注册，需要进行如下验证，请在JS中使用正则表达式进行验证。

1. 用户名：只能由英文字母和数字组成，长度为4～16个字符，并且以英文字母开头
2. 密码： 大小写字母和数字6-20个字符
3. 确认密码：两次密码要相同
4. 电子邮箱: 符合邮箱地址的格式
5. 手机号：1开头，3/5/6/8为第二个数字，其他为数字开头，长度11个字符
6. 生日：生日的年份在1900～2009之间，生日格式为1980-5-12或1988-05-04的形式



案例分析：

1. 创建正则表达式
2. 得到文本框中输入的值
3. 如果不匹配，在后面的span中显示错误信息，返回false
4. 如果匹配，在后面的span中显示一个打勾图片，返回true
5. 写一个验证表单中所有的项的方法，所有的方法都返回true，这个方法才返回true.

案例代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>验证注册页面</title>

    <script src="js/public.js"></script>
    <style type="text/css">
        span {
            /*出错信息为红色*/
            color: red;
        }

        body {
            margin: 0;
            padding: 0;
            font-size: 12px;
            line-height: 20px;
        }
        .main {
            width: 525px;
            margin-left: auto;
            margin-right: auto;
        }


        .left {
```

```
            text-align: right;
            width: 80px;
            height: 25px;
            padding-right: 5px;
        }
        .center {
            width: 280px;
        }
        .in {
            width: 130px;
            height: 16px;
            border: solid 1px #79abea;
        }

        div {
            color: #F00;
        }
    </style>
    <script type="text/javascript">
        //验证所有的项
        function checkAll() {
            //所有的方法都返回true才表示通过
            return checkUser() && checkPassword() && checkEmail()&& checkMobile()&&
checkBirthday();
        }

        //只能由英文字母和数字组成，长度为4～16个字符，并且以英文字母开头
        function checkUser() {


              var reg=/^[a-zA-Z]\w{3,15}$/;

            if(reg.test(getEle("user").value)){
                getEle("userInfo").innerHTML="";
                return true;
            }else{
                getEle("userInfo").innerHTML="error";
                return false;
            }

        }
        // 大小写字母和数字6-20个字符
        function checkPassword() {
            var reg=/^\w{6,20}$/;

            if(reg.test(getEle("pwd").value)){
                getEle("pasInfo").innerHTML="";
                return true;
            }else{
                getEle("pasInfo").innerHTML="error";
                return false;


            }
```

```
        }

        //验证邮箱
        function checkEmail() {
            // zhangsan@163.com
            //lisi@164.cn
            var reg=/^\w+@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$/;

            if(reg.test(getEle("email").value)){
                getEle("emailInfo").innerHTML="";
                return true;
            }else{
                getEle("emailInfo").innerHTML="error";
                return false;

            }
        }

        //验证手机号
        function checkMobile() {
            //
            var reg=/^1[3|5|7|8]\d{9}$/;

            if(reg.test(getEle("mobile").value)){
                getEle("mobileInfo").innerHTML="";
                return true;
            }else{
                getEle("mobileInfo").innerHTML="error";
                return false;

            }
        }

        //验证生日
        function checkBirthday() {

            //1. /^(190-9-([0-9]|1-2)-(0-3))|(20-9[0-9]-([0-9]|1-2)-(0-3))$/

            var reg=/^(19[0-9][0-9]|200[0-9])-([1-9]|0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1]|
[1-9])$/;

            if(reg.test(getEle("birth").value)){
                getEle("birthdayInfo").innerHTML="";
                return true;
            }else{
                getEle("birthdayInfo").innerHTML="error";
                return false;

            }
        }

    </script>

</head>
```

```
<body>
<form action="server" method="post" id="myform" onsubmit="return checkAll()">
    <table class="main" border="0" cellspacing="0" cellpadding="0">

        <tr>
            <td><h3>新用户注册</h3></td>
        </tr>
        <tr>
            <td style="height:10px;"></td>
        </tr>
        <tr>
            <td>
                <table width="100%" border="0" cellspacing="0" cellpadding="0">
                    <tr>
                        <!-- 不能为空， 输入长度必须介于 5 和 10 之间 -->
                        <td class="left">用户名：</td>
                        <td class="center">
                            <!--失去焦点就调用-->
                            <input id="user" name="user" type="text" class="in"
onblur="checkUser()"/>
                            <span id="userInfo"></span>
                        </td>
                    </tr>
                    <tr>
                        <!-- 不能为空， 输入长度大于6个字符 -->
                        <td class="left">密码：</td>
                        <td class="center">
                            <input id="pwd" name="pwd" type="password" class="in"
onblur="checkPassword()"/>
                            <span id="pasInfo"></span>
                        </td>
                    </tr>
                    <tr>
                        <!-- 不能为空， 与密码相同 -->
                        <td class="left">确认密码：</td>
                        <td class="center">
                            <input id="repwd" name="repwd" type="password" class="in"/>
                        </td>
                    </tr>
                    <tr>
                        <!-- 不能为空， 邮箱格式要正确 -->
                        <td class="left">电子邮箱：</td>
                        <td class="center">
                            <input id="email" name="email" type="text" class="in"
onblur="checkEmail()"/>
                            <span id="emailInfo"></span>
                        </td>
                    </tr>
                    <tr>
                        <!-- 不能为空， 使用正则表达式自定义校验规则,1开头，11位全是数字 -->
                        <td class="left">手机号码：</td>

                        <td class="center">
```

```
                            <input id="mobile" name="mobile" type="text" class="in"
onblur="checkMobile()"/>
                            <span id="mobileInfo"></span>
                        </td>
                    </tr>
                    <tr>
                        <!-- 不能为空， 要正确的日期格式 -->
                        <td class="left">生日：</td>
                        <td class="center">
                            <input id="birth" name="birth" type="text" class="in"
onblur="checkBirthday()"/>
                            <span id="birthdayInfo"></span>
                        </td>
                    </tr>
                    <tr>
                        <td class="left"> </td>
                        <td class="center">
                            <input name="" type="submit" style="width: 100px;height: 30px;"/>
                        </td>
                    </tr>
                </table></td>
        </tr>
    </table>
</form>
</body>
</html>
```

内置对象：日期对象

创建 Date 对象的语法：

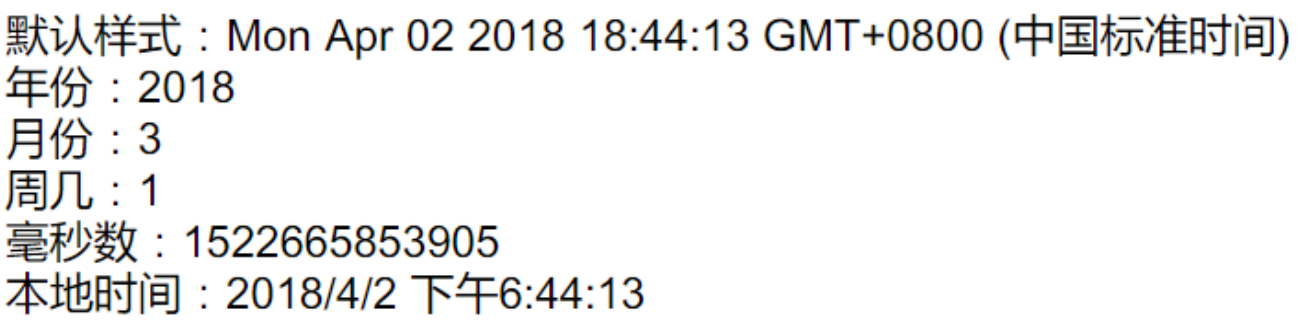
var 变量名 = new Date()
说明：创建一个当前的时间和日期对象

日期对象的方法

方法名	作用
getFullYear()	从 Date 对象以四位数字返回年份。
getMonth()	从 Date 对象返回月份 (0 ~ 11)。
getDate()	从 Date 对象返回一个月中的某一天 (1 ~ 31)。
getDay()	从 Date 对象返回一周中的某一天 (0 ~ 6)。其中：0表示周日，1~6周一到周六
getHours()	返回 Date 对象的小时 (0 ~ 23)。
getMinutes()	返回 Date 对象的分钟 (0 ~ 59)。
getSeconds()	返回 Date 对象的秒数 (0 ~ 59)。
getMilliseconds()	返回 Date 对象的毫秒(0 ~ 999)。
getTime()	返回 1970 年 1 月 1 日至今的毫秒数。类似于Java中的System.currentTimeMillis()
toLocaleString()	根据本地时间格式，把 Date 对象转换为字符串。

日期对象的方法演示

- 运行效果：



- 案例代码：

```
<script type="text/javascript">
    var date = new Date();
    document.write("默认样式: " + date +  "<br/>");
    document.write("年份: " + date.getFullYear() +  "<br/>");
    document.write("月份: " + (date.getMonth() + 1) +  "<br/>");
    document.write("周几: " + date.getDay()  +  "<br/>");
    document.write("毫秒数: " + date.getTime()  +  "<br/>");
    document.write("本地时间: " + date.toLocaleString()  +  "<br/>");
</script>
```

BOM编程

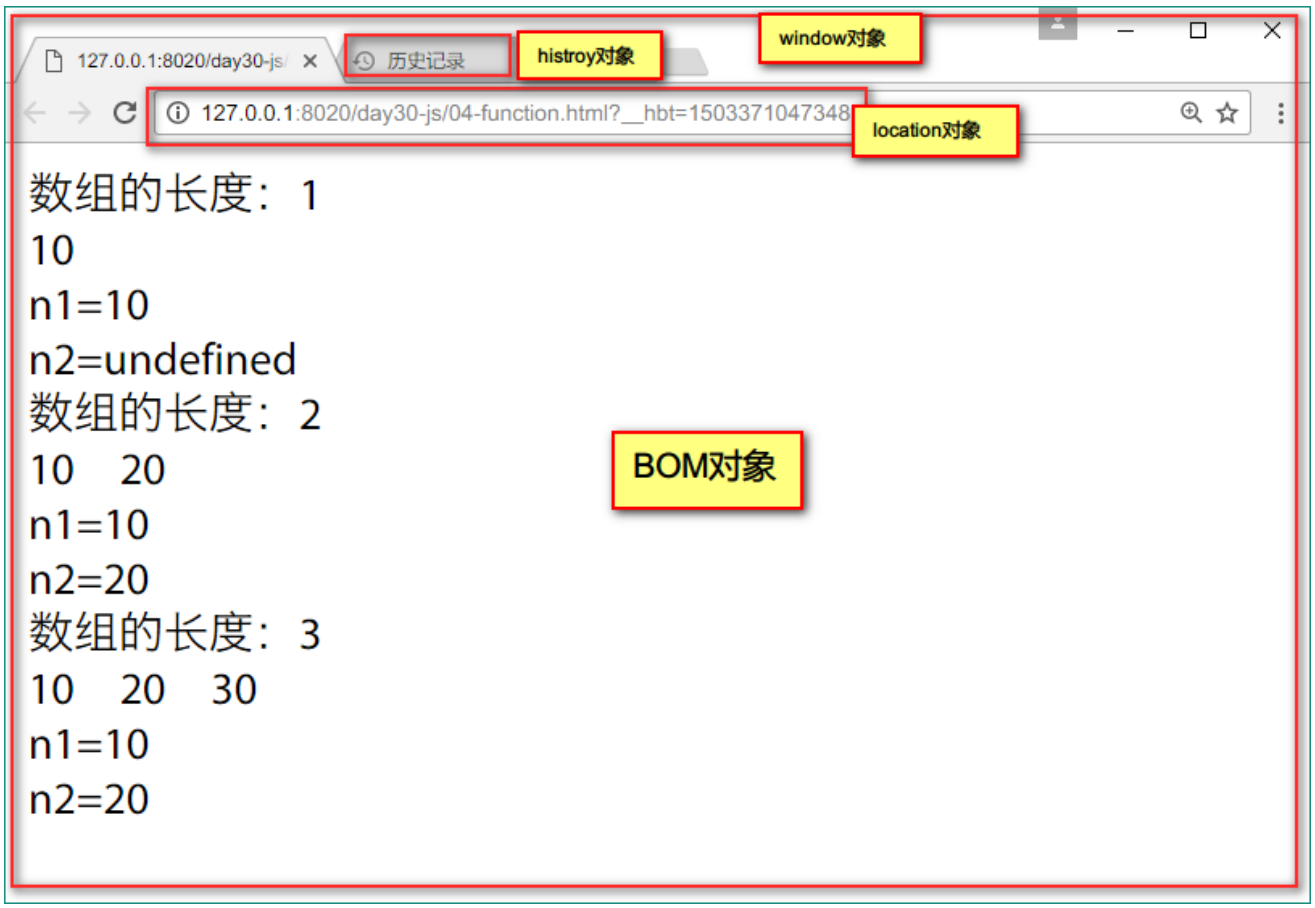
BOM编程概述

Browser Object Model 浏览器对象模型

将浏览器的各个组成部分封装成对象。

BOM编程的作用

作用：对浏览器中的各种对象进行操作



BOM常用的对象

BOM常用对象	作用
window	表示浏览器窗体对象
location	浏览器地址栏对象
history	访问历史记录对象

location对象

作用：用于页面跳转

location常用的属性

href属性	作用
获取href属性	得到当前浏览器访问地址
设置href属性	就可以进行页面跳转，跳转到新的页面

```
<script type="text/javascript">
    //得到当前访问地址
    document.write(location.href + "<br/>");
    //设置属性值，页面进行跳转
    location.href = "http://www.baidu.cn";
</script>
```

location常用的方法

location的方法	描述
reload()	重新加载当前页面，相当于刷新功能

```
<script type="text/javascript">
    //得到当前访问地址
    document.write(location.href + "<br/>");
    //设置属性值，页面进行跳转
    //location.href = "http://www.baidu.cn";

    //输出时间
    document.write(new Date().toLocaleString() + "<br/>");
</script>
<input type="button" onclick="location.reload()" value="刷新">
```

history对象

作用：代表浏览器访问过的历史记录，可以通过这个对象访问之前访问过的页面

方法

方法	作用
forward()	相当浏览器上前进按钮
back()	相当于浏览器上后退按钮
go(正数或负数)	正数相当于前进，负数相当于后退。1前进或后退1个页面

- 注：浏览器上的前进和后退按钮可以点的时候，这个代码才起作用。
- 案例演示：

```
<input type="button" onclick="history.forward()" value="前进">

<input type="button" value="后退" onclick="history.back()">
```

window中与计时有关的方法

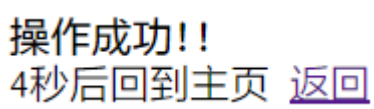
window中的方法	作用
setInterval("函数名()", 间隔毫秒数) setInterval(函数名,毫秒数)	每过一段时间执行1次函数，调用无限次 时间单位：毫秒 返回值： 整数，当前的计时器
clearInterval(计时器)	用于清除上面这个方法创建计时器，让计时器停止。
setTimeout("函数名()", 间隔毫秒数) setInterval(函数名,毫秒数)	过一段时间以后执行1次函数，调用1次 返回值： 整数，当前的计时器
clearTimeout(计时器)	用于清除上面这个方法创建计时器

案例：倒计时跳转到另一个页面

案例需求：

页面上显示一个倒计时5秒的数字，到了5秒以后跳转到另一个页面

案例效果：



案例分析：

1. 在页面上创建一个span用于放置变化的数字。
2. 定义一个全局变量为5，每过1秒调用1次refresh()函数
3. 编写refresh()函数，修改span中的数字

4. 判断变量是否为1，如果是1则跳转到新的页面
5. 否则变量减1。并修改span中的数字

案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<span id="time">5</span>秒后回到主页
<script type="text/javascript">
    //创建一个用于计数的全局变量
    var count = 4;

    //每过1秒调用一次函数
    window.setInterval(function(){
        //修改time中数字
        document.getElementById("time").innerHTML = count;
        //如果等于1，进行页面的跳转
        if (count == 0) {
            location.href = "http://www.baidu.cn";
        }
        //减一
        count --;
    }, 1000);
</script>
</body>
</html>
```

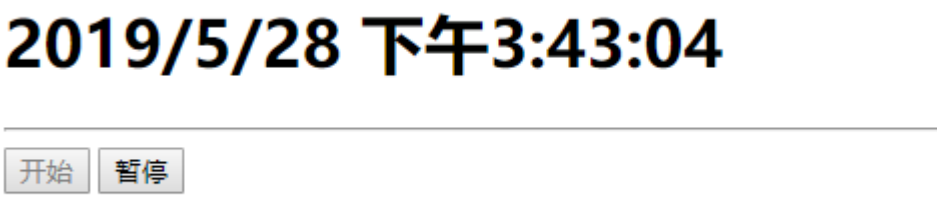
案例：会动的时钟

案例需求：

页面上有两个按钮，一个开始按钮，一个暂停按钮。点开始按钮时间开始走动，点暂停按钮，时间不动。再点开始按钮，时间继续走动。点开始按钮，按钮就不可用。点暂停按钮，开始按钮才可以使用。

- 设置disabled属性设置为true不可用

案例效果：



案例分析：

1. 在页面上创建一个h1标签，用于显示时钟，设置颜色和大小。
2. 点开始按钮调用一个方法start()，在方法内部每过1秒中调用另一个方法begin()
3. begin()方法内部得到现在的时间，并将得到的时间显示在h1标签内部
4. 暂停的按钮调用另一个方法：pause()，在方法内部清除clearInterval()的计时器。
5. 为了防止多次点开始按钮出现bug，在开始调用计时器之前清除上一个计时器。
6. 也可以将开始按钮点击以后设置为不可用，点暂停按钮的时候，开始按钮再次可用。

案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1 id="clock"></h1>
<hr>
<input type="button" value="开始" onclick="begin()" id="begin">
<input type="button" value="暂停" onclick="pause()" id="pause" disabled="disabled">

<script type="text/javascript">
    //创建全局变量
    var timer = 0;

    //开始按钮
    function begin() {
        //每过1秒设置时间到h1中，返回值就是计时器
        timer = window.setInterval("start()", 1000);
        //设置按钮不可用
        document.getElementById("begin").disabled = true;
        //暂停按钮设置为可用
        document.getElementById("pause").disabled = false;
    }

    //开始时间
    function start() {
        document.getElementById("clock").innerHTML = new Date().toLocaleString();  //设置现在的时
间
    }
```

```
    //暂停按钮
    function pause() {
        //清除计时器
        window.clearInterval(timer);
        //暂停按钮不可用
        document.getElementById("pause").disabled = true;
        //开始按钮设置为可用
        document.getElementById("begin").disabled = false;
    }

</script>
</body>
</html>
```

innerHTML和innerText的区别

元素的属性	作用
innerHTML	获得： 得到某个元素内部的HTML内容 设置： 修改某个元素内部的HTML内容
innerText	获得： 得到某个元素内部纯文本内容 设置： 修改某个元素内部的纯文本

- 案例需求：

通过这个案例学习innerHTML和innerText的区别

- 案例效果：

- 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>innerHTML和innerText</title>
</head>
<body>
<div id="myDiv">
    <span>
        <a href="#">我是链接</a>

    </span>
```

```
</div>
<hr/>
<!--innerHTML和innerText区别-->
<input type="button" id="b1" value="得到HTML">
<input type="button" id="b2" value="设置HTML">
<input type="button" id="b3" value="得到Text">
<input type="button" id="b4" value="设置Text">

<script type="text/javascript">
    //得到HTML
    document.getElementById("b1").onclick = function () {
        alert(document.getElementById("myDiv").innerHTML);
    }

    //设置HTML
    document.getElementById("b2").onclick = function () {
        document.getElementById("myDiv").innerHTML = "<h2 style='color: red'>我是标题</h2>"
    }

    //得到Text
    document.getElementById("b3").onclick = function () {
        alert(document.getElementById("myDiv").innerText);
    }

    //设置text
    document.getElementById("b4").onclick = function () {
        document.getElementById("myDiv").innerText = "<h2 style='color: red'>我是标题</h2>"
    }
</script>
</body>
</html>
```

与对话框有关的方法

window中与对话框有关的方法	作用
alert("提示信息")	弹出一个信息框
string prompt("提示信息","默认值")	弹出一个输入框
boolean confirm("提示信息")	弹出确认框，有确认和取消两个按钮，如果点击确认，返回true，如果点击取消，返回的是false

- confirm演示案例：

```
<input type="button" value="删除" id="del">
```

```
<script type="text/javascript">
    document.getElementById("del").onclick = function () {
        //如果为真，表示点确认
        if (window.confirm("真的要删除吗?")) {
            alert("已经成功删除");
        }
        else {
            alert("取消删除");
        }
    }
</script>
```

window中方法小结:

window中的方法	说明
setInterval()	每过1段时间调用一次函数
setTimeout()	过1段时间以后调用1次函数，只调用1次
clearTimeout() / clearInterval()	清除上面的两个计时器
alert()/prompt()/confirm()	信息框 输入框 确认框

DOM编程

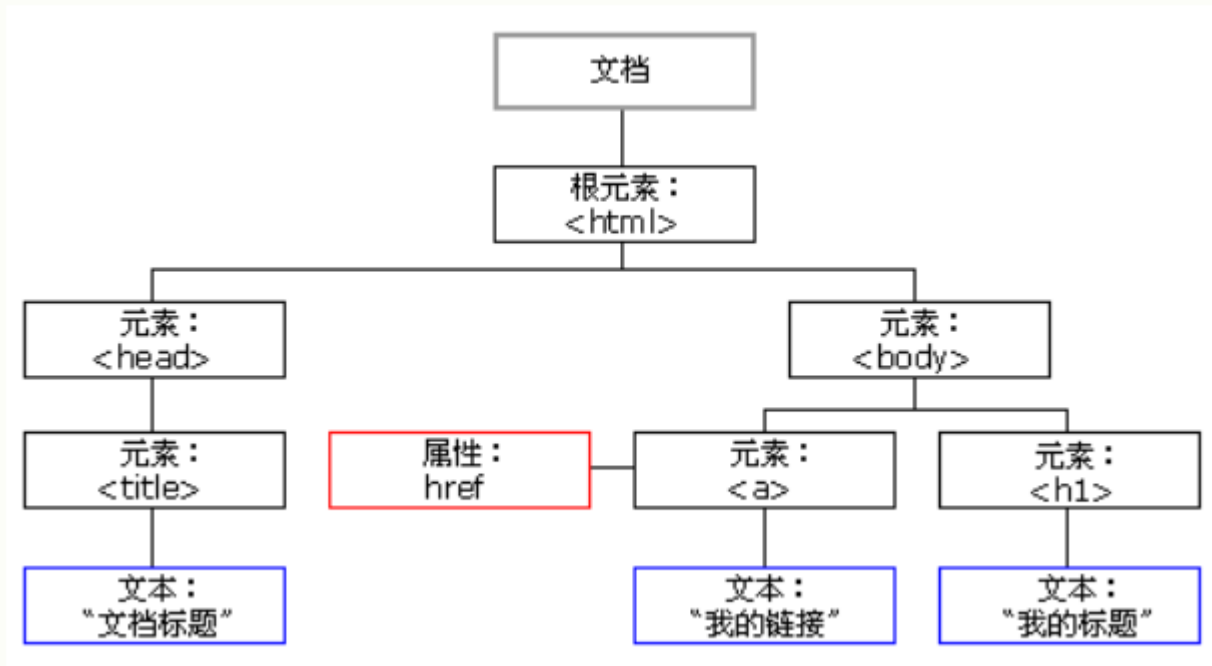
DOM编程的基本概念

Document Object Model 文档对象模型，整个网页就是一个文档对象，可以dom来操作网页中所有的标签元素。每标签还有属性，内容，都可以进行查询或修改。达到修改网页结构目的，看到网页内容会动态发生变化。

DOM编程的作用

每个HTML页面在被浏览器解析的时候都会在内存中创建一棵DOM树，我们通过编写JS代码就可以访问这棵树上任何一个节点，并且对节点进行操作。通过DOM模型，可以访问所有的 HTML元素，连同它们所包含的文本和属性。可以对其中的内容进行修改和删除，同时也可以创建新的元素。新创建的元素对象，要挂到DOM树上才可以在网页上显示出来。

- **节点与DOM树**



查找节点

查找节点的方法

获取元素的方法	作用
document.getElementById("id")	通过id得到唯一元素
document.getElementsByName("name")	通过标签的name属性得到一组名字相同的元素
document.getElementsByTagName ("标签名")	通过标签的名字得到一组相同的标签，如：input
document.getElementsByClassName("类名")	通过标签class类名属性，得到一组标签

案例：查找节点：

- 案例需求：

学习使用上面的几个方法，点击第1个按钮给所有的a链接添加href属性；点击第2个按钮，给div内部添加

- 案例分析：

1. 在window.onload加载事件中给三个按钮添加点击事件
2. 第1组是a标签，通过标签名得到这组元素，遍历每个元素，给它的href属性赋值地址
3. 第2组是div，通过标签名得到这组元素，使用innerHTML添加a标签为子元素
4. 第3组是div，通过类名得到这组元素，使用innerHTML添加a标签为子元素

- 案例效果：

- 案例代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>根据标签的属性找元素</title>
</head>
<body>
<input type="button" value="(通过标签名)给a链接添加地址" id="b2"/>
<input type="button" value="(通过name属性)给div设值" id="b3"/>
<input type="button" value="(通过类名)给div设值" id="b4"/>
<hr/>
<a>京东</a><br/>
<a>京东</a><br/>
<a>京东</a><br/>
<hr/>
<div name="one"></div>
<div name="one"></div>
<div name="one"></div>
<hr/>
<div class="two"></div>
<div class="two"></div>
<div class="two"></div>
</body>

<script type="text/javascript">
    //点b2按钮设置a标签的href属性
    document.getElementById("b2").onclick = function () {
        //通过标签的名字得到所有的a标签
        var arr = document.getElementsByTagName("a");
        //循环得到每个元素

        for (var i = 0; i < arr.length; i++) {
```

```
            //添加href属性
            arr[i].href = "http://www.jd.com";
        }
    }

    document.getElementById("b3").onclick = function () {
        //通过name得到一组元素
        var arr = document.getElementsByName("one");
        //相当于forEach功能，obj表示每个元素
        for (var obj of arr) {
            obj.innerHTML = "<a href='http://www.baidu.com'>百度</a>";
        }
    }

    document.getElementById("b4").onclick = function () {
        //通过类名得到一组元素
        var arr = document.getElementsByClassName("two");
        for (var i = 0; i < arr.length; i++) {
            arr[i].innerHTML = "<a href='http://www.baidu.com'>阿里云</a>";
        }
    }

</script>
</html>
```

DOM树的修改

> 添加网页的元素，修改DOM树分2步：

1. 创建网页元素
2. 将创建好的元素挂到DOM树上面去

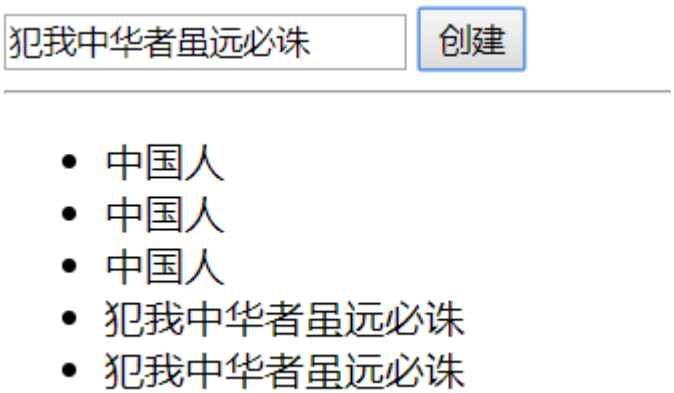
创建元素的方法

创建元素的方法	作用
document.createElement("标签名")	创建一个元素 参数：标签名字，如：td

修改DOM树的方法

将元素挂到DOM树上的方法	作用
父元素.appendChild(子元素)	父元素是DOM树中已经存在元素 子元素是新创建的元素，把创建好的子元素挂到DOM树上，做为父元素的子元素。
父元素.removeChild(子元素)	将DOM树上某个元素的子元素删除
元素.remove()	自己删除本身这个元素

案例：




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Demo</title>
</head>

<body>
<input type="text" id="txt">
<input type="button" id="btn" value="创建">
<hr>
<ul id="uid">


</ul>
</body>
<script>

    var uObj=document.getElementById("uid");

    document.getElementById("btn").onclick=function (ev) {
        var liObj=document.createElement("li");
```

```
        liObj.innerHTML=document.getElementById("txt").value;

        uObj.appendChild(liObj);

    }

</script>

</html>
```

通过关系找节点

遍历节点的属性	说明
childNodes	返回当前元素所有的子元素
firstChild	得到当前元素的第1个子节点
lastChild	得到当前元素的最后1个子节点
parentNode	得到当前元素的父节点
nextSibling	得到当前元素的下xue一个兄弟节点
previousSibling	得到当前元素的上一个兄弟节点

案例：学生信息管理

案例需求：

1. 使用CSS：当鼠标移入时，该行的背景颜色为黄色，当鼠标移出时，该行的背景颜色还原；
2. 当添加按钮“添加一行数据”时，文本框中的数据添加到表格中且文本框置空；
3. 当点击表格中的“删除”时，该行数据被删除，删除前确认

案例效果：

学号	姓名	操作
001	张三	删除
002	李四	删除
003	王五	删除

学号： 003 姓名： 王五 添加

案例分析：

1. 添加的实现：当点击按钮时，得到文本框中的文本，之后将文本框的值清空。

2. 创建tr节点；把整个tr中的内容使用innerHTML直接设置到tr的内部。
3. 把tr追加到tbody元素中；注：tbody无论源代码中是否写了，在浏览器中始终存在。
4. 删除操作：将当前点击的a标签所在的一行tr，从tbody中删除。
5. 如何得到a标签所在的行？先得到a标签的父节点td，td的父节点tr，然后删除。
6. href="javascript:;"是让链接不进行跳转，只执行onclick中指定的方法。

案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>学生信息管理</title>
    <style type="text/css">
        th{
            background-color: yellow;
        }
        tr:hover{
            background-color: pink;
        }
        table{
            text-align: center;
        }
    </style>
</head>
<body>
<table border="1" cellspacing="0" width="500px" align="center" id="info">
    <tr>
        <th>学号</th>
        <th>姓名</th>
        <th>操作</th>
    </tr>
</table>
<hr>
<div style="text-align: center">
学号：<input type="text" placeholder="请输入学号" id="num">
姓名：<input type="text" placeholder="请输入姓名" id="name1">
<input type="button" value="添加" onclick="add()">
</div>
<script src="js/public.js"></script>
<script type="text/javascript">
    var Tinfo = getEle("info");//获取表格对象
    function add() {
        var numb = getEle("num").value;//获取学号内容
        var name1 = getEle("name1").value;//获取姓名内容

        //创建tr  td
        var tr1 = document.createElement("tr");
        var td1 = document.createElement("td");
        var td2 = document.createElement("td");
        var td3 = document.createElement("td");
```

```
        td1.innerHTML=numb;
        td2.innerHTML=name1;
        td3.innerHTML="<a href='#' onclick='remove(this)'>删除</a>"

        /*把td挂载到tr树上面*/
        tr1.appendChild(td1);
        tr1.appendChild(td2);
        tr1.appendChild(td3);

        /*把tr添加到table树*/
        Tinfo.appendChild(tr1);
    }
    function remove(obj) {
        /*根据点击的a标签找到对应的父标签(td)再找到对应的父标签(tr)*/
        obj.parentNode.parentNode.remove();
    }
</script>
</body>
</html>
```