JavaScript 提高

第1节　学习目标

1) 能够使用正则表达式进行表单的验证
2) 能够使用 window 对象常用的方法
3) 能够使用 location 对象常用的方法和属性
4) 能够使用 history 对象常用的方法
5) 能够使用 DOM 中来查找节点
6) 能够使用 DOM 来增删改节点
7) 能够使用 JavaScript 对 CSS 样式进行操作

第2节　正则对象

2.1　创建的方式

2.1.1 方式 1：

正则表达式是 JS 中是一个类：RegExp = Regular Expression 正则表达式

```
var reg = new RegExp("正则表达式");
```

2.1.2 方式 2：

以/开头，以/结尾，中间的部分就是正则表达式

```
var reg = /正则表达式/;
```

2.1.3 两种方式的区别：

1. 在 js 中，正则表达式的两种声明方式对于“\d、\D”之类的匹配模式中，前者需要转义，而后者无需转义
2. 前者支持字符串拼接，支持变量，更加灵活；后者对于固定的表达式，书写起来方便快捷、更加直观。

2.1.4 匹配模式：

i 忽略大小写进行比较，两种写法：

```
var reg = new RegExp("正则表达式", "匹配模式");
var reg = /正则表达式/匹配模式;
```

2.2　常用的方法

JS 中正则表达式的方法　说明	
boolean test("字符串")	**如果正则表达式匹配字符串，返回 true，否则返回 false**

2.2.1 基本使用示例：

```
//方式一：RegExp
var reg = new RegExp("\\d{3}");

//方式二：/正则表达式/
var reg = /\d{3}/;

//判断是否匹配
var flag = reg.test("123");
```

```
//ignore 忽略
var reg = new RegExp("cat","i");

var reg = /cat/i;
var flag = reg.test("CAT");
document.write("结果: " + flag);
```

2.3 JS 匹配上与 Java 中的不同

Java 默认情况下必须要精确匹配，而在 JS 中默认是模糊匹配，只要字符串包含了正则表达式的内容就返回 true

正则表达式	匹配字符串	Java 中匹配结果	JavaScript 中匹配结果
\d{3}	a123b	false	**true**
^\d{3}	123b	false	**true**
\d{3}$	a123	false	**true**
^\d{3}$	**123**	**true**	**true**

2.4 案例：校验表单

2.4.1 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 /^\w+@\w+(.[a-zA-Z]{2,3}){1,2}$/
5. 手机号：/^1[34578]\d{9}$/
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，/^((19\d{2})|(200\d))-(0?[1-9]|1[0-2])-(0?[1-9]|[1-2]\d|3[0-1])$/

2.4.2 案例效果：

2.4.3 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.

2.4.4 案例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>验证注册页面</title>
<style type="text/css">
body {
    margin: 0;
    padding: 0;
    font-size: 12px;
    line-height: 20px;
}
.main {
    width: 525px;
    margin-left: auto;
    margin-right: auto;
}
.hr_1 {
    font-size: 14px;
    font-weight: bold;
    color: #3275c3;
    height: 35px;
    border-bottom-width: 2px;
    border-bottom-style: solid;
    border-bottom-color: #3275c3;
    vertical-align: bottom;
    padding-left: 12px;
}
.left {
    text-align: right;
    width: 80px;
    height: 25px;
    padding-right: 5px;
}
.center {
    width: 280px;
}
.in {
    width: 130px;
    height: 16px;
    border: solid 1px #79abea;
}

.red {
    color: #cc0000;
    font-weight: bold;
}

div {
    color: #F00;
}
```

```
</style>
<script type="text/javascript">
//验证表单中所有的项
function checkAll () {
    //所有的方法都返回 true，这个方法才返回 true
    return checkUser() && checkMail();
}

//验证用户名
function checkUser () {
    //1. 创建正则表达式
    var reg = /^[a-zA-Z][a-zA-Z0-9]{3,15}$/;
    //2. 得到文本框中输入的值
    var value = document.getElementById("user").value;
    //3. 如果不匹配，在后面的 span 中显示错误信息，返回 false
    if (reg.test(value)==false) {
        document.getElementById("userInfo").innerHTML = "用户名不正确";
        return false;
    }
    //4. 如果匹配，在后面的 span 中显示一个打勾图片，返回 true
    else {
        document.getElementById("userInfo").innerHTML = "<img src='img/gou.png' width='15'/>";
        return true;
    }
}

//验证邮箱
function checkMail () {
    //1. 创建正则表达式
    var reg = /^\w+@\w+(\.[a-zA-Z]{2,3}){1,2}$/;
    //2. 得到文本框中输入的值
    var value = document.getElementById("email").value;
    //3. 如果不匹配，在后面的 span 中显示错误信息，返回 false
    if (reg.test(value)==false) {
        document.getElementById("emailInfo").innerHTML = "邮箱格式不正确";
        return false;
    }
    //4. 如果匹配，在后面的 span 中显示一个打勾图片，返回 true
    else {
        document.getElementById("emailInfo").innerHTML = "<img src='img/gou.png' width='15'/>";
        return true;
    }
}
</script>
</head>

<body>
<form action="server" method="post" id="myform" onsubmit="return checkAll()">
<table class="main" border="0" cellspacing="0" cellpadding="0">
    <tr>
        <td><img src="img/logo.jpg" alt="logo" /><img src="img/banner.jpg" alt="banner" /></td>
    </tr>
    <tr>
        <td class="hr_1">新用户注册</td>
    </tr>
```

```
<tr>
    <td style="height:10px;"></td>
</tr>
<tr>
    <td>
    <table width="100%" border="0" cellspacing="0" cellpadding="0">
        <tr>
            <!-- 长度为 4 ~ 16 个字符，并且以英文字母开头 -->
            <td class="left">用户名：</td>
            <td class="center">
            <input id="user" name="user" type="text" class="in" onblur="checkUser()"/>
            <span style="color: red" 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" />
            </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="checkMail()"/>
            <span id="emailInfo" style="color: red;"></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"/>
            </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"/>
            </td>
        </tr>
        <tr>
            <td class="left"> </td>
            <td class="center">
            <input name="" type="image" src="img/register.jpg" />
            </td>
        </tr>
```

```
        </table></td>
    </tr>
</table>
</form>
</body>
</html>
```

第3节　BOM 编程

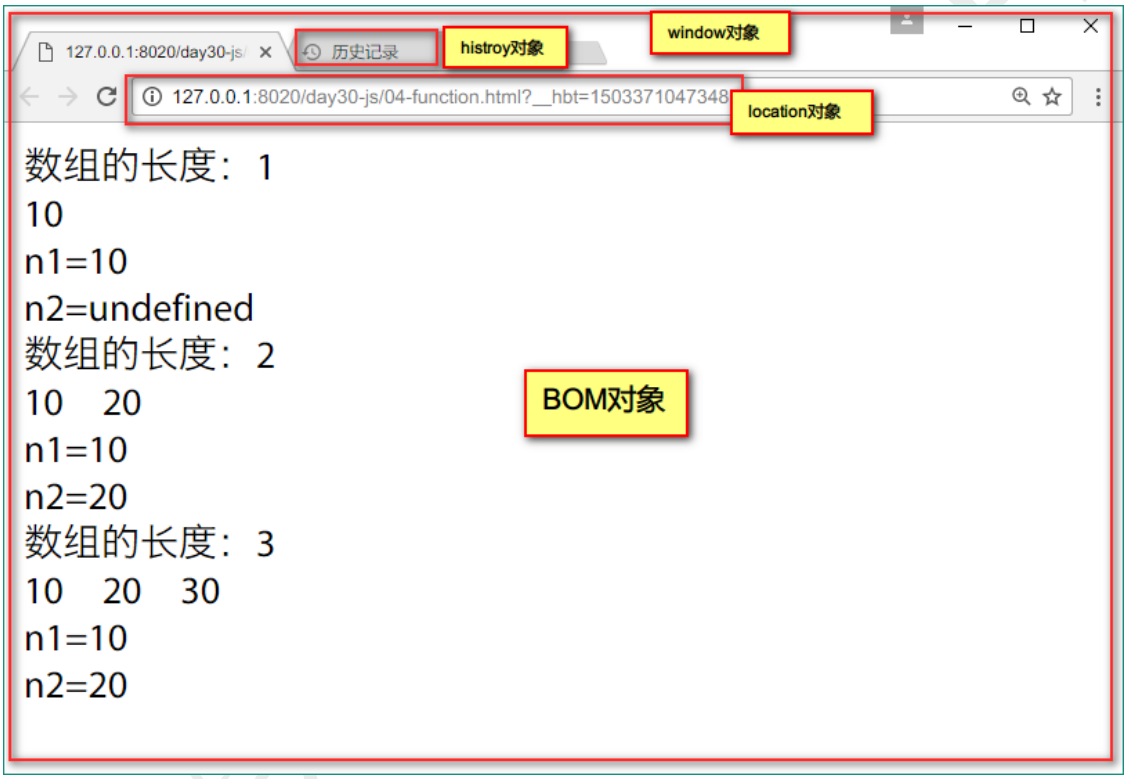
3.1　BOM 编程概述

3.1.1 BOM 编程的概念

BOM：Browser Object Model 浏览器对象模型

3.1.2 BOM 编程的作用

用于操作浏览器中的各种对象，不同浏览器定义上是有差别的，实现方式也会有不同。以下是 Chrome 浏览器各个 BOM 对象。



3.1.3 BOM 常用的对象

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

3.2　window 对象

BOM 的核心对象是 window，它表示浏览器的一个实例。

注:只要是 window 的方法和属性，window 对象名都可以省略

3.2.1 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”)	弹出一个确认按钮的信息框
string prompt(“提示信息”,"默认值”)	弹出一个输入信息框，返回字符串类型
boolean confirm("提示信息")	弹出一个信息框，有确定和取消按钮。如果点确定，返回 true，点取消返回 false

3.2.2 与计时有关的方法

window 中的方法	作用
setTimeout(函数名,间隔毫秒数)	在指定的时间后调用 1 次函数，只执行 1 次，单位是毫秒。返回值：返回一个整数类型的计时器函数调用有两种写法：1) setTimeout("函数名(参数)", 1000);2) setTimeout(函数名,1000, 参数); 注意方式二：没有引号，没有括号
setInterval(函数名,间隔毫秒数)	每过指定的时间调用 1 次函数，不停的调用函数，单位是毫秒。返回值：返回一个整数类型的计时器。
clearInterval(计时器)	清除 setInterval()方法创建的计时器
clearTimeout(计时器)	清除 setTimeout 创建的计时器

3.2.3 修改元素内容的几个方法和属性

名称	作用
方法：document.getElementById("id")	通过 id 得到一个元素，如果有同名的元素则得到第 1 个
属性：innerHTML	获得：元素内部的 HTML 设置：修改元素内部的 HTML
属性：innerText	获得：元素内部的文本 设置：修改元素内部的纯文本，其中的 html 标签不起作用

3.3 案例：会动的时钟

3.3.1 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。

3.3.2 案例效果：

3.3.3 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()，在方法内部清除上面 setInterval()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。

3.3.4 案例代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>会动的时钟</title>
    <style type="text/css">
        #clock {
            color: green;
            font-size: 30px;
        }
    </style>
</head>
<body>
    <script type="text/javascript">
        var timer;

        //开始调用
        function start () {
            //先清除上一个计时器，再开启一个计时器
            window.clearInterval(timer);
            //1000 毫秒调用 begin()
            timer = window.setInterval("begin()", 1000);
        }

        //思路：每过 1 秒钟调用 1 次时间，并且将时间显示在某个元素内部
        function begin () {
            //得到现在的时间
            var time = new Date();
            //得到 h1 元素
            var clock = document.getElementById("clock");
            //将时间显示在 h1 中
            clock.innerHTML = time.toLocaleString();
        }

        //暂停
        function pause () {
            //清除计时器
            window.clearInterval(timer);
        }
    </script>

    <h1 id="clock">我是时间</h1>
    <input type="button" value="开始" onclick="start()" />
    <input type="button" value="暂停" onclick="pause()" />
```

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

3.4 location 对象

3.4.1 location 是什么

代表浏览器的地址栏对象

3.4.2 location 常用的属性

href 属性	作用
获取 href 属性	获得当前地址栏访问的地址
设置 href 属性	用于页面的跳转，跳转到一个新的页面

3.4.3 location 常用的方法

location 的方法	描述
reload()	重新加载当前的页面，相当于浏览器上的刷新按钮

3.4.4 代码的演示

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>location 对象</title>
    </head>
    <body>
    <input type="button" value="跳转到传智" onclick="jump()" />
    <script type="text/javascript">
        //获得属性
        //document.write("获得 href 属性: " + location.href + "<br/>");
        //设置属性
        function jump () {
            location.href = "http://www.itcast.cn";
        }
    </script>
    </body>
</html>
```

3.5 history 对象

3.5.1 作用

访问浏览器之前已经访问过的页面

3.5.2 方法

方法	作用
forward()	类似于浏览器上前进按钮

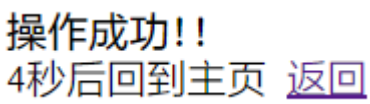
back()	类似于浏览器上后退按钮
go()	正数或负数，go(1)相当于 forward()，go(-1)相当于 back()

3.6 案例：倒计时跳转到另一个页面

3.6.1 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

3.6.2 案例效果：



3.6.3 案例分析：

1. 页面上创建一个 span 用于显示变化的数字，点返回链接直接跳转。
2. 定义一个变量为 5，每过 1 秒调用 1 次刷新 refresh()函数
3. 编写函数，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

3.6.4 案例代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
    <span id="time">5</span>秒后回到主页
    <script type="text/javascript">
        //定义一个变量为 5
        var count = 5;

        //一开始就执行
        window.setInterval("refresh()", 1000);

        function refresh() {
            if (count > 1 ) {
                //修改 span 中的数字
                document.getElementById("time").innerText = --count;
            }
            else {
                location.href = "http://www.baidu.com";
            }
        }
    </script>
</body>
</html>
```

第4节 DOM 编程

4.1 DOM 编程概述

4.1.1 DOM 编程的基本概念

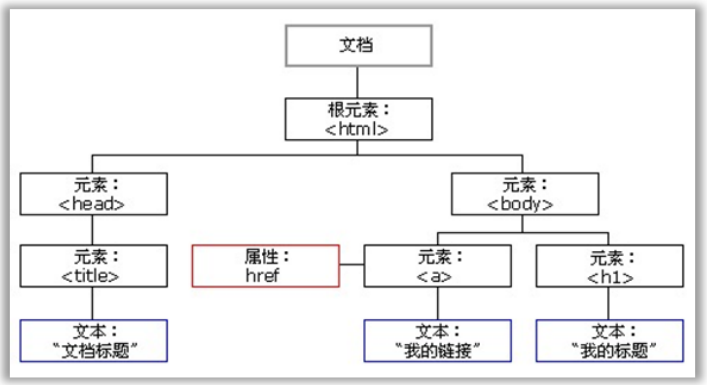
Document Object Model 文档对象模型，用于操作网页中元素

4.1.2 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。

```
<html>
  <head>
    <title>文档标题</title>
  </head>
  <body>
    <a href="">我的链接</a>
    <h1>我的标题</h1>
  </body>
</html>
```

DOM 树



4.2 查找节点

4.2.1 查找节点的方法

获取元素的方法	作用
document.getElementById("id")	通过 id 属性到唯一的元素如果页面上有多个同名的 id，则得到第 1 个元素
document.getElementsByName("name")	通过 name 属性得到一组标签，返回一个数组
document.getElementsByTagName ("标签名")	通过标签名字得到一组标签，返回一个数组
document.getElementsByClassName("类名")	通过类名得到一组标签，返回一个数组

4.3 案例：查找节点：

4.3.1 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

4.3.2 案例效果：

4.3.3 案例代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>根据标签的属性找元素</title>
    <script type="text/javascript">
        window.onload = function () {
            //根据标签名找元素
            var b2 = document.getElementById("b2");
            b2.onclick = function () {
                //返回的是一个数组对象
                var aNodes = document.getElementsByTagName("a");
                for (var index = 0; index < aNodes.length; index++) {
                    aNodes[index].href = "http://www.itcast.cn";
                }
            }

            //根据 name 的属性值找
            var b3 = document.getElementById("b3");
            b3.onclick = function () {
                //根据 name 的属性值找元素，返回一个数组对象
                var divs = document.getElementsByName("one");
                for (var index = 0; index < divs.length; index++) {
                    divs[index].innerHTML = "<a href='#'>黑马程序员</a>";
                }
            }

            var b4 = document.getElementById("b4");
            b4.onclick = function () {
                //根据 class 的属性值找元素，返回一个数组对象
                var divs = document.getElementsByClassName("two")
                for (var index = 0; index < divs.length; index++) {
                    divs[index].innerHTML = "<a href='#'>JavaEE 开发</a>";
                }
```

```
            }
        }
    </script>
</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>传智播客</a><br/>
    <a>传智播客</a><br/>
    <a>传智播客</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>
</html>
```

4.4 案例：实现全选/全不选，商品结算的功能

4.4.1 案例需求：

页面上有 5 件商品，前面都有复选框，名字叫 item，value 是商品的价格。下面有一个"全选/全不选"的复选框，id 是"all"，点它实现全选或全不选的功能，还有个反选的按钮，点它实现反选的功能。下面有一个按钮，点它则计算选中商品的总金额。

4.4.2 案例效果：

4.4.3 案例分析：

1. 知识点：复选框如果要选中，设置 checked=true，取消设置 checked=false。
2. 全选：通过 name 属性得到上面所有的复选框对象，遍历集合，将每一个元素的 checked 设置为 true。
3. 全不选：将所有元素的 checked 属性设置为 false。
4. 反选：原来选中的设置 false，原来没选的设置为 true。
5. 结账：将所有选中的元素的 value 转成数字，再累加，将累加的结果显示在后面的 span 中。

4.4.4 案例代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
    <script type="text/javascript">
        //全选/全不选
        function selectAll () {
            //得到下面复选框的状态
            var all = document.getElementById("all");
            //得到上面所有的复选框
            var items = document.getElementsByName("item");
            for (var i = 0; i < items.length; i++) {
                items[i].checked = all.checked;
            }
        }

        //反选
        function reverseSelect () {
            //得到上面所有的复选框
            var items = document.getElementsByName("item");
            for (var i = 0; i < items.length; i++) {
                items[i].checked = !items[i].checked;
            }
        }

        //结账
        function total() {
            var sum = 0;
            //得到上面所有的复选框
            var items = document.getElementsByName("item");
            for (var i = 0; i < items.length; i++) {
                //选中的才加
                if (items[i].checked) {
                    //把值相加
                    sum+=parseFloat(items[i].value);
                }
            }
            //显示到 span 中
            document.getElementById("result").innerHTML = "￥" + sum;
        }
    </script>
</head>
<body>
    <h3>商品价格列表</h3>
    <input type="checkbox" name="item" value="1500" /> 山地自行车 1500<br />
    <input type="checkbox" name="item" value="200" /> 时尚女装 200<br />
    <input type="checkbox" name="item" value="3000" /> 笔记本电脑 3000 元<br />
    <input type="checkbox" name="item" value="800" /> 情侣手表 800<br />
    <input type="checkbox" name="item" value="2000" /> 桑塔纳 2000<br />
    <hr/>
    <input type="checkbox" id="all" onclick="selectAll()" />全选/全不选  
    <input type="button" id="reverse" onclick="reverseSelect()" value=" 反 选 "/> 
```

```
    <input type="button" value=" 结 账 " onclick="total()" /> <span id="result"></span>
</body>
</html>
```

4.5 增删改节点

在 DOM 树上创建元素分 2 步：

1. 创建元素
2. 将元素挂到 DOM 树上

4.5.1 创建和修改元素的方法

创建元素的方法	作用
document.createElement("标签名")	在文档上创建一个元素对象
元素对象.setAttribute("属性名", "属性值")	给元素添加一个属性名和属性值 如果元素名不存在则是添加属性，存在则是修改属性值
document.createTextNode("文本内容")	在文档上创建一个文本节点

4.5.2 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)	将元素追加成父元素的最后一个子元素
父元素.removeChild(子元素)	通过父元素删除一个子元素
元素.remove()	元素删除本身

4.5.3 通过关系找节点

遍历节点的属性	说明
childNodes	得到当前元素下所有的子节点
firstChild	得到当前元素的第一个子节点
lastChild	得到当前元素的最后一个子节点
parentNode	得到当前元素的父节点
nextSibling	得到当前元素的下一个兄弟节点
previousSibling	得到当前元素的上一个兄弟节点

4.6 案例：学生信息管理

4.6.1 案例需求：

1. 使用 CSS 样式：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

4.6.2 案例效果：

学号	姓名	操作
00001	潘金莲	删除
00002	鲁智深	删除

学号：83931 姓名：武松 添加

4.6.3 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. 创建文本节点，创建 td，创建 tr；把 tr 追加到 tbody 元素中；td 追加到 tr 中；文本节点追加到 td 中；注：tbody 无论源代码中是否写了，在浏览器中始终存在。
3. 删除链接那个单元格使用 innerHTML 来实现，这是另一种简单的实现方式。
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。

4.6.4 案例代码：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>学生信息管理</title>
<style type="text/css">
    table {
        width: 500px;
    }

    th {
        background-color: lightgray;
    }

    /*伪类样式*/
    tr:hover {
        background-color: yellow;
    }
</style>
<script type="text/javascript">
    //添加一行
    function addRow () {
        //得到文本框中的值
        var no = document.getElementById("no").value;

        //清空
        document.getElementById("no").value = "";

        //创建文本对象
        var txtNo= document.createTextNode(no);
        //创建 td1
        var td1 = document.createElement("td");
        td1.appendChild(txtNo);
```

```
        //创建 tr
        var tr = document.createElement("tr");
        tr.appendChild(td1);

        //添加第 2 格
        var name = document.getElementById("name").value;
        document.getElementById("name").value = "";
        //创建文本对象
        var nameText= document.createTextNode(name);
        //创建 td2
        var td2 = document.createElement("td");
        td2.appendChild(nameText);
        //直接 td2 挂到 tr 上
        tr.appendChild(td2);

        //简单做法，直接修改 td 的 innerHTML
        var td3 = document.createElement("td");
        td3.innerHTML = "<a href=\"javascript:void(0)\" onclick=\"deleteRow(this)\">删除</a>";

        tr.appendChild(td3);

        //得到 tbody 对象
        var tbody = document.getElementById("data");

        tbody.appendChild(tr);
    }

    //删除一行
    function deleteRow (obj) {
        if (confirm("真的要删除这个学生吗?")) {
            //得到 a 标签的父元素的父元素
            obj.parentNode.parentNode.remove();
        }
    }
</script>
</head>
<body>
<div>
<table border="1" cellspacing="0" cellpadding="3">
    <tbody id="data">
    <tr>
        <th>学号</th>
        <th>姓名</th>
        <th>操作</th>
    </tr>
    <tr>
        <td>00001</td>
        <td>潘金莲</td>
        <td>
            <a href="javascript:;" onclick="deleteRow(this)">删除</a>
        </td>
    </tr>
    <tr>
        <td>00002</td>
```

```
        <td>鲁智深</td>
        <td>
            <a href="javascript:;" onclick="deleteRow(this)">删除</a>
        </td>
    </tr>
    </tbody>
</table>
<br />
学号: <input type="text" id="no" value="" />
姓名: <input type="text" id="name" value="" />
<input type="button" id="addBtn" value="添加" onclick="addRow()"/>
</div>
</body>
</html>
```

第5节　操作 CSS 样式

5.1　在 JS 中操作 CSS 属性命名上的区别

以前 css 直接写死在 html 中，现在可以通过 js 脚本去动态修改一个标签的样式。

CSS 中写法	JS 中的写法	说明
color	color	**一个单词的样式写法是相同**
font-size	**fontSize**	**驼峰命名法，首字母小写，第二个单词以后首字母大写**

5.1.1 方式一：

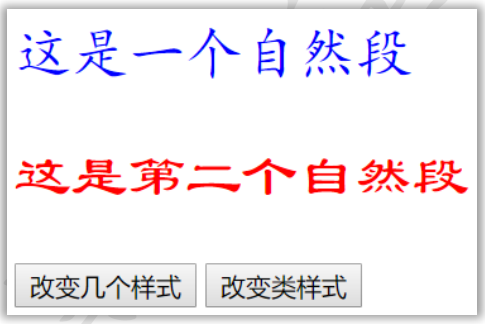
```
元素.style.样式名 = "样式值";
```

5.1.2 方式二：

```
元素.className = "类名";
```

5.2　示例：点按钮，修改 p 标签的字体、颜色、大小

5.2.1 效果



5.2.2 代码：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
```

```
            .myclass {
                font-family: "隶书";
                color: red;
                font-size: 40px;
            }
        </style>
        <script type="text/javascript">
            //修改样式
            function changeCss () {
                var p = document.getElementById("first");
                //字体、颜色、大小
                p.style.fontFamily = "楷体";
                p.style.color="blue";
                p.style.fontSize = "40px";
            }

            //修改类样式
            function changeClass () {
                var p = document.getElementById("second");
                //修改类样式
                p.className = "myclass";
            }
        </script>
    </head>
    <body>
        <p id="first">
            这是一个自然段
        </p>
        <p id="second">
            这是一个自然段
        </p>
        <input type="button"  value="改变几个样式" onclick="changeCss()"/>
        <input type="button"  value="改变类样式" onclick="changeClass()" />
    </body>
</html>
```

5.2.3 网页的变化

```
<p id="first" style="font-family: 楷体; color: blue; font-size: 40px;">
            这是一个自然段
        </p>
<p id="second" class="myclass">
            这是一个自然段
        </p>
```

5.3 案例：使用 JS 修改表格行的背景色

5.3.1 案例需求：

使用 JS 修改表格行的背景色，产生隔行变色的效果，鼠标移上去的时候这一行变成其它颜色，移出去的时候还原成之前的背景色。

5.3.2 案例效果：

分类ID	分类名称	分类描述	操作
1	手机数码	手机数码类商品	修改|删除
2	电脑办公	电脑办公类商品	修改|删除
3	鞋靴箱包	鞋靴箱包类商品	修改|删除
4	家居饰品	家居饰品类商品	修改|删除

5.3.3 案例分析：

1. 创建三个类样式，分别用于设置背景色为浅红，浅黄，浅绿。
2. 通过标签名得到所有的 tr 行
3. 在窗体加载完毕的事件中遍历所有的行，如果是偶数，则设置它的背景色为浅黄色，否则为浅红色
4. 设置鼠标在上面和在外面的事件，鼠标在上面的事件中，记录没有换颜色之前的颜色，再设置类样式为浅绿色。设置一个全局变量记录之前的类样式名。
5. 如果鼠标移出之后，要回到原来的颜色，将全局变量记录的样式名赋值给当前行的类样式名。

5.3.4 案例代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
    <style type="text/css">
        table {
            margin: auto;
            border-collapse: collapse;
        }

        tr {
            text-align: center;
            height: 32px;
        }

        .redStyle {
            background: lightpink;
        }

        .yellowStyle {
            background: lightyellow;
        }

        .greenStyle {
            background: lightgreen;
        }
    </style>
```

```
    <script type="text/javascript">
        //记录颜色
        var color = "";

        window.onload = function () {
            //获取所有行
            var trNodes = document.getElementsByTagName("tr");
            //遍历所有的行，如果是偶数，则设置为浅黄色
            for (var index = 1; index < trNodes.length; index++) {
                if (index % 2 == 0) {
                    trNodes[index].className = "yellowStyle";
                } else {
                    trNodes[index].className = "redStyle";
                }

                //鼠标经过的事件
                trNodes[index].onmouseover = function () {
                    //记录没有换颜色之前的颜色
                    color = this.className;
                    this.className = "greenStyle";
                }

                //鼠标移出事件
                trNodes[index].onmouseout = function () {
                    //如果鼠标移出之后，要回到原来的颜色。
                    this.className = color;
                }
            }
        }
    </script>
</head>
<body>
<table id="tab1" border="1" width="800" align="center">
    <tr style="background-color: #ccc;">
        <th>分类 ID</th>
        <th>分类名称</th>
        <th>分类描述</th>
        <th>操作</th>
    </tr>
    <tr>
        <td>1</td>
        <td>手机数码</td>
        <td>手机数码类商品</td>
        <td><a href="">修改</a>|<a href="">删除</a></td>
    </tr>
    <tr>
        <td>2</td>
        <td>电脑办公</td>
        <td>电脑办公类商品</td>
        <td><a href="">修改</a>|<a href="">删除</a></td>
    </tr>
    <tr>
        <td>3</td>
        <td>鞋靴箱包</td>
        <td>鞋靴箱包类商品</td>
```

```
        <td><a href="">修改</a>|<a href="">删除</a></td>
    </tr>
    <tr>
        <td>4</td>
        <td>家居饰品</td>
        <td>家居饰品类商品</td>
        <td><a href="">修改</a>|<a href="">删除</a></td>
    </tr>
</table>
</body>
</html>
```