JavaScript

1.1 HTML中嵌入JavaScript代码的三种方式

HTML中嵌入JavaScript代码的第一种方式：

```
<!doctype html>
<html>
    <head>
        <title>HTML中嵌入JS代码的第一种方式</title>
    </head>
    <body>

        <!--
            1、要实现的功能：
                用户点击以下按钮，弹出消息框。

            2、JS是一门事件驱动型的编程语言，依靠事件去驱动，然后执行对应的程序。
            在JS中有很多事件，其中有一个事件叫做：鼠标单击，单词：click。并且任何
            事件都会对应一个事件句柄叫做：onclick。【注意：事件和事件句柄的区别是：
            事件句柄是在事件单词前添加一个on。】，而事件句柄是以HTML标签的属性存在
            的。

            3、onclick="js代码"，执行原理是什么？
                页面打开的时候，js代码并不会执行，只是把这段JS代码注册到按钮的click事件上
了。
                等这个按钮发生click事件之后，注册在onclick后面的js代码会被浏览器自动调用。

            4、怎么使用JS代码弹出消息框？
                在JS中有一个内置的对象叫做window，全部小写，可以直接拿来使用，window代表的
是浏览器对象。
                window对象有一个函数叫做:alert，用法是：window.alert("消息");这样就可以
弹窗了。

            5、JS中的字符串可以使用双引号，也可以使用单引号。

            6、JS中的一条语句结束之后可以使用分号“;”，也可以不用。
        -->
        <input type="button" value="hello" onclick="window.alert('hello js')"/>

        <input type="button" value="hello" onclick='window.alert("hello
jscode")'/>

        <input type="button" value="hello" onclick="window.alert('hello
zhangsan')
                                                    window.alert('hello lis')
                                                    window.alert('hello
wangwu')"/>

        <!-- window. 可以省略。-->
        <input type="button" value="hello" onclick="alert('hello zhangsan')
                                                    alert('hello lis')
                                                    alert('hello wangwu')"/>
```

```
        <input type="button" value="hello" onclick="alert('hello zhangsan');
                                                    alert('hello lis');
                                                    alert('hello wangwu');"/>

    </body>
</html>
```

HTML中嵌入JavaScript代码的第二种方式：

```
<!--
    javascript的脚本块在一个页面当中可以出现多次。没有要求。
    javascript的脚本块出现位置也没有要求，随意。
-->
<script type="text/javascript">
// alert有阻塞当前页面加载的作用。（阻挡，直到用户点击确定按钮。）
window.alert("first.......");
</script>

<!doctype html>
<html>
    <head>
        <title>HTML中嵌入JS代码的第二种方式</title>

        <!--样式块-->
        <style type="text/css">
            /*
                css代码
            */
        </style>

        <script type="text/javascript">
            window.alert("head.............");
        </script>

    </head>
    <body>

        <input type="button" value="我是一个按钮对象1" />

        <!--第二种方式：脚本块的方式-->
        <script type="text/javascript">

            /*
                暴露在脚本块当中的程序，在页面打开的时候执行，
                并且遵守自上而下的顺序依次逐行执行。（这个代
                码的执行不需要事件）
            */
            window.alert("Hello World!"); // alert函数会阻塞整个HTML页面的加载。

            // JS代码的注释，这是单行注释。
            /*
                JS代码的多行注释。和java一样。
            */
            window.alert("Hello JavaScript!");
```

```
        </script>

        <input type="button" value="我是一个按钮对象" />

    </body>
</html>

<script type="text/javascript">
window.alert("last.......");
</script>

<!--
/**
*
* javadoc注释，这里的注释信息会被javadoc.exe工具解析提取生成帮助文档。
*/
-->
```

HTML中嵌入JavaScript代码的第三种方式：

```
<!doctype html>
<html>
    <head>
        <title>HTML中嵌入JS代码的第三种方式：引入外部独立的js文件。</title>
    </head>
    <body>

        <!--在需要的位置引入js脚本文件-->
        <!--引入外部独立的js文件的时候，js文件中的代码会遵循自上而下的顺序依次逐行执行。-->
        <!--
        <script type="text/javascript" src="js/1.js"></script>
        -->

        <!--同一个js文件可以被引入多次。但实际开发中这种需求很少。-->
        <!--
        <script type="text/javascript" src="js/1.js"></script>
        -->

        <!--这种方式不行，结束的script标签必须有。-->
        <!--
        <script type="text/javascript" src="js/1.js" />
        -->
        <!--
        <script type="text/javascript" src="js/1.js"></script>
        -->

        <script type="text/javascript" src="js/1.js">
            // 这里写的代码不会执行。
            // window.alert("Test");
        </script>

        <script type="text/javascript">
            alert("hello jack!");
        </script>


    </body>
```

```
</html>
```

1.2 BOM

001-BOM编程-open和close.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>BOM编程-open和close</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、BOM编程中，window对象是顶级对象，代表浏览器窗口。
                2、window有open和close方法，可以开启窗口和关闭窗口。
            */

        </script>

        <input type="button" value="开启百度(新窗口)"
onclick="window.open('http://www.baidu.com');" />
        <input type="button" value="开启百度(当前窗口)"
onclick="window.open('http://www.baidu.com', '_self');" />
        <input type="button" value="开启百度(新窗口)"
onclick="window.open('http://www.baidu.com', '_blank');" />
        <input type="button" value="开启百度(父窗口)"
onclick="window.open('http://www.baidu.com', '_parent');" />
        <input type="button" value="开启百度(顶级窗口)"
onclick="window.open('http://www.baidu.com', '_top');" />

        <input type="button" value="打开表单验证"  onclick="window.open('002-
open.html')"/>
    </body>
</html>
```

002-open.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>close</title>
    </head>
    <body>
        <input type="button" value="关闭当前窗口" onclick="window.close();" />
    </body>
</html>
```

003-弹出消息框和确认框.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>弹出消息框和确认框</title>
    </head>
    <body>
        <script type="text/javascript">
            function del(){
                /*
                var ok = window.confirm("亲，确认删除数据吗？");
                //alert(ok);
                if(ok){
                    alert("delete data ....");
                }
                */
                if(window.confirm("亲，确认删除数据吗？")){
                    alert("delete data ....");
                }
            }
        </script>
        <input type="button" value="弹出消息框" onclick="window.alert('消息框!')"
/>

        <!--删除操作的时候都要提前先得到用户的确认。-->
        <input type="button" value="弹出确认框(删除)" onclick="del();" />
    </body>
</html>
```

004-当前窗口设置为顶级窗口.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>当前窗口设置为顶级窗口</title>
        <!--窗体-->
        <!-- <frameset cols="30%,*">
            <frame src="http://www.baidu.com" />
            <frame src="005-child-window.html" />
        </frameset> -->
    </head>
    <body>

        <!--在当前窗口中隐藏的内部窗体。-->
        <!-- <iframe src="http://www.baidu.com"></iframe> -->

        <iframe src="005-child-window.html"></iframe>

    </body>
</html>
```

005-child-window.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>child-window</title>
    </head>
    <body>
        child window.
        <script type="text/javascript">
            window.onload = function(){
                var btn = document.getElementById("btn");
                btn.onclick = function(){
                    if(window.top != window.self){
                        //window.top = window.self;
                        window.top.location = window.self.location;
                    }
                }
            }
        </script>
        <input type="button" value="将当前窗口设置为顶级窗口" id="btn" />
    </body>
</html>
```

006-history对象.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>history对象</title>
    </head>
    <body>
        <a href="007.html">007页面</a>
        <input type="button" value="前进" onclick="window.history.go(1)"/>
    </body>
</html>
```

007.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>007</title>
    </head>
    <body>
        007 page!
        <input type="button" value="后退" onclick="window.history.back()" />
        <input type="button" value="后退" onclick="window.history.go(-1)" />
    </body>
</html>
```

008-设置浏览器地址栏上的URL.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>设置浏览器地址栏上的URL</title>
    </head>
    <body>

        <script type="text/javascript">
            function goBaidu(){
                //var locationObj = window.location;
                //locationObj.href = "http://www.baidu.com";

                // window.location.href = "http://www.jd.com";
                // window.location = "http://www.126.com";

                //document.location.href = "http://www.sina.com.cn";
                document.location = "http://www.tmall.com";
            }
        </script>

        <input type="button" value="新浪" onclick="goBaidu();"/>

        <input type="button" value="baidu"
onclick="window.open('http://www.baidu.com');" />

    </body>
</html>

<!--
    总结，有哪些方法可以通过浏览器往服务器发请求？
        1、表单form的提交。
        2、超链接。<a href="http://localhost:8080/oa/save?
username=zhangsan&password=123">用户只能点击这个超链接</a>
        3、document.location
        4、window.location
        5、window.open("url")
        6、直接在浏览器地址栏上输入URL，然后回车。（这个也可以手动输入，提交数据也可以成为动态
的。）
```

```
        以上所有的请求方式均可以携带数据给服务器，只有通过表单提交的数据才是动态的。
-->
```

1.3 DOM

001-DOM编程-获取文本框的value.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>DOM编程-获取文本框的value</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、JavaScript包括三大块：
                    ECMAScript：JS的核心语法（ES规范 / ECMA-262标准）
                    DOM：Document Object Model（文档对象模型：对网页当中的节点进行增删改的过程。）HTML文档被当做一棵DOM树来看待。
                        var domObj = document.getElementById("id");
                    BOM：Browser Object Model（浏览器对象模型）
                        关闭浏览器窗口、打开一个新的浏览器窗口、后退、前进、浏览器地址栏上的地址等，都是BOM编程。
                2、DOM和BOM的区别和联系？
                    BOM的顶级对象是：window
                    DOM的顶级对象是：document
                    实际上BOM是包括DOM的！
             */
            /*
            window.onload = function(){
                //var btnElt = window.document.getElementById("btn");
                var btnElt = document.getElementById("btn");
                alert(btnElt); // object HTMLInputElement
            }
            */

          window.onload = function(){
              var btnElt = document.getElementById("btn");
              btnElt.onclick = function(){
                  /*
                  // 获取username节点
                  var usernameElt = document.getElementById("username");
                  var username = usernameElt.value;
                  alert(username);
                  */
                 // alert(document.getElementById("username").value);

                 // 可以修改它的value
                 document.getElementById("username").value = "zhangsan";
              }
          }
        </script>

        <!--
        <input type="button" id="btn" value="hello" />
```

```
        -->

        <input type="text" id="username" />
        <input type="button" value="获取文本框的value" id="btn"/>

        <hr>

        <script type="text/javascript">
            window.onload = function(){
                document.getElementById("setBtn").onclick = function(){
                    document.getElementById("username2").value =
document.getElementById("username1").value;
                }
            }
        </script>

        <input type="text" id="username1" />
        <br>
        <input type="text" id="username2" />
        <br>
        <input type="button" value="将第一个文本框中的value赋值到第二个文本框上"
id="setBtn" />

        <!--blur事件：失去焦点事件-->
        <!--以下代码中的this代表的是当前input节点对象,this.value就是这个节点对象的value属
性。-->
        <input type="text" onblur="alert(this.value)" />

    </body>
</html>
```

002-DOM编程-innerHTML和innerText操作div和span.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>DOM编程-innerHTML和innerText操作div和span</title>
        <style type="text/css">
            #div1{
                background-color: aquamarine;
                width: 300px;
                height: 300px;
                border: 1px black solid;
                position: absolute;
                top: 100px;
                left: 100px;
            }
        </style>
    </head>
    <body>

        <!--
            innerText和innerHTML属性有什么区别？
                相同点：都是设置元素内部的内容。
                不同点：
```

```
                    innerHTML会把后面的“字符串”当做一段HTML代码解释并执行。
                    innerText，即使后面是一段HTML代码，也只是将其当做普通的字符串来看待。
        -->
        <script type="text/javascript">
            window.onload = function(){
                var btn = document.getElementById("btn");
                btn.onclick = function(){
                    // 设置div的内容
                    // 第一步:获取div对象
                    var divElt = document.getElementById("div1");
                    // 第二步:使用innerHTML属性来设置元素内部的内容
                    // divElt.innerHTML = "fjdkslajfkdlsajkfldsjaklfds";
                    // divElt.innerHTML = "<font color='red'>用户名不能为空!
</font>";
                    divElt.innerText = "<font color='red'>用户名不能为空! </font>";
                }
            }
        </script>

        <input type="button" value="设置div中的内容" id="btn"/>

        <div id="div1"></div>

    </body>
</html>
```

003-DOM编程-关于正则表达式.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>DOM编程-关于正则表达式</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、什么是正则表达式，有什么用？
                    正则表达式：Regular Expression
                    正则表达式主要用在字符串格式匹配方面。

                2、正则表达式实际上是一门独立的学科，在Java语言中支持，C语言中也支持，
javascript中也支持。
                大部分编程语言都支持正则表达式。正则表达式最初使用在医学方面，用来表示神经符号
等。目前使用最多
                的是计算机编程领域，用作字符串格式匹配。包括搜索方面等。

                3、正则表达式，对于我们javascript编程来说，掌握哪些内容呢？
                    第一：常见的正则表达式符号要认识。
                    第二：简单的正则表达式要会写。
                    第三： 他人编写的正则表达式要能看懂。
                    第四：在javascript当中，怎么创建正则表达式对象！（new对象）
                    第五：在javascript当中，正则表达式对象有哪些方法！（调方法）
                    第六：要能够快速的从网络上找到自己需要的正则表达式。并且测试其有效性。

                4、常见的正则表达式符号？
```

```
                    . 匹配除换行符以外的任意字符
                    \w 匹配字母或数字或下划线或汉字
                    \s 匹配任意的空白符
                    \d 匹配数字
                    \b 匹配单词的开始或结束
                    ^ 匹配字符串的开始
                    $ 匹配字符串的结束

                    * 重复零次或更多次
                    + 重复一次或更多次
                    ? 重复零次或一次
                    {n} 重复n次
                    {n,} 重复n次或更多次
                    {n,m} 重复n到m次

                    \W 匹配任意不是字母，数字，下划线，汉字的字符
                    \S 匹配任意不是空白符的字符
                    \D 匹配任意非数字的字符
                    \B 匹配不是单词开头或结束的位置
                    [^x] 匹配除了x以外的任意字符
                    [^aeiou] 匹配除了aeiou这几个字母以外的任意字符

                    正则表达式当中的小括号()优先级较高。
                    [1-9] 表示1到9的任意1个数字（次数是1次。）
                    [A-Za-z0-9] 表示A-Za-z0-9中的任意1个字符
                    [A-Za-z0-9-] 表示A-Z、a-z、0-9、- ，以上所有字符中的任意1个字符。

                    | 表示或者

                5、简单的正则表达式要会写
                    QQ号的正则表达式：^[1-9][0-9]{4,}$

                6、他人编写的正则表达式要能看懂？
                    email正则：^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$

                7、怎么创建正则表达式对象，怎么调用正则表达式对象的方法？
                    第一种创建方式：
                        var regExp = /正则表达式/flags;
                    第二种创建方式:使用内置支持类RegExp
                        var regExp = new RegExp("正则表达式","flags");

                    关于flags：
                        g：全局匹配
                        i：忽略大小写
                        m：多行搜索（ES规范制定之后才支持m。）当前面是正则表达式的时候，m不
能用。只有前面是普通字符串的时候，m才可以使用。

                    正则表达式对象的test()方法？
                        true / false = 正则表达式对象.test(用户填写的字符串);
                        true : 字符串格式匹配成功
                        false: 字符串格式匹配失败
            */
            window.onload = function(){
                // 给按钮绑定click
                document.getElementById("btn").onclick = function(){
                    var email = document.getElementById("email").value;
                    var emailRegExp = /^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+
([-.]\w+)*$/;
```

```
                    var ok = emailRegExp.test(email);
                    if(ok){
                         //合法
                         document.getElementById("emailError").innerText = "邮箱地
址合法";
                    }else{
                        // 不合法
                        document.getElementById("emailError").innerText = "邮箱地址
不合法";
                    }
                }
                // 给文本框绑定focus
                document.getElementById("email").onfocus = function(){
                    document.getElementById("emailError").innerText = "";
                }
            }

        </script>

        <input type="text" id="email" />
        <span id="emailError" style="color: red; font-size: 12px;"></span>
        <br>
        <input type="button" value="验证邮箱" id="btn" />
    </body>
</html>
```

004-去除字符串的前后空白trim.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>去除字符串的前后空白trim</title>
    </head>
    <body>
        <script type="text/javascript">
            // 低版本的IE浏览器不支持字符串的trim()函数,怎么办?
            // 可以自己对String类扩展一个全新的trim()函数!
            String.prototype.trim = function(){
                // alert("扩展之后的trim方法");
                // 去除当前字符串的前后空白
                // 在当前的方法中的this代表的就是当前字符串.
                //return this.replace(/^\s+/, "").replace(/\s+$/, "");
                return this.replace(/^\s+|\s+$/g, "");
            }

            window.onload = function(){
                document.getElementById("btn").onclick = function(){
                    // 获取用户名
                    var username = document.getElementById("username").value;
                    // 去除前后空白
                    username = username.trim();
                    // 测试
                    alert("--->" + username + "<----");
                }
            }
```

```
        </script>
        <input type="text" id="username" />
        <input type="button" value="获取用户名" id="btn" />
    </body>
</html>

<!--
表单验证：
    （1）用户名不能为空
    （2）用户名必须在6-14位之间
    （3）用户名只能有数字和字母组成，不能含有其它符号（正则表达式）
    （4）密码和确认密码一致，邮箱地址合法。
    （5）统一失去焦点验证
    （6）错误提示信息统一在span标签中提示，并且要求字体12号，红色。
    （7）文本框再次获得焦点后，清空错误提示信息，如果文本框中数据不合法要求清空文本框的value
    （8）最终表单中所有项均合法方可提交
-->
```

005-表单验证.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>表单验证</title>
        <style type="text/css">
            span {
                color: red;
                font-size: 12px;
            }
        </style>
    </head>
    <body>
        <script type="text/javascript">
            /*
            （1）用户名不能为空
            （2）用户名必须在6-14位之间
            （3）用户名只能有数字和字母组成，不能含有其它符号（正则表达式）
            （4）密码和确认密码一致，邮箱地址合法。
            （5）统一失去焦点验证
            （6）错误提示信息统一在span标签中提示，并且要求字体12号，红色。
            （7）文本框再次获得焦点后，清空错误提示信息，如果文本框中数据不合法要求清空文本框
的value
            （8）最终表单中所有项均合法方可提交
            */
           window.onload = function(){
               // 获取username的span标签
               var usernameErrorSpan = document.getElementById("usernameError");
               // 给用户名文本框绑定blur事件
               var usernameElt = document.getElementById("username");
               usernameElt.onblur = function(){
                   // 获取用户名
                   var username = usernameElt.value;
                   // 去除前后空白
                   username = username.trim();
                   // 判断用户名是否为空
                   /*
```

```
                    if(username){
                        // 代表username不是空字符串
                        alert("username = " + username);
                    }else{
                        // 代表username是空字符串
                        alert("username是空字符串");
                    }
                    */
                    // if(username.length == 0){}
                    if(username === ""){
                        // 用户名为空
                        usernameErrorSpan.innerText = "用户名不能为空";
                    }else{
                        // 用户名不为空
                        // 继续判断长度[6-14]
                        if(username.length < 6 || username.length > 14){
                            // 用户名长度非法
                            usernameErrorSpan.innerText = "用户名长度必须在[6-14]之
间";
                        }else{
                            // 用户名长度合法
                            // 继续判断是否含有特殊符号
                            var regExp = /^[A-Za-z0-9]+$/;
                            var ok = regExp.test(username);
                            if(ok){
                                // 用户名最终合法
                            }else{
                                // 用户名中含有特殊符号
                                usernameErrorSpan.innerText = "用户名只能由数字和字母
组成";
                            }
                        }
                    }
                }

                // 给username这个文本框绑定获得焦点事件
                usernameElt.onfocus = function(){
                    // 清空非法的value
                    if(usernameErrorSpan.innerText != ""){
                        usernameElt.value = "";
                    }
                    // 清空span
                    usernameErrorSpan.innerText = "";
                }

                // 获取密码错误提示的span标签
                var pwdErrorSpan = document.getElementById("pwdError");
                // 获取确认密码框对象
                var userpwd2Elt = document.getElementById("userpwd2");
                // 绑定blur事件
                userpwd2Elt.onblur = function(){
                    // 获取密码和确认密码
                    var userpwdElt = document.getElementById("userpwd");
                    var userpwd = userpwdElt.value;
                    var userpwd2 = userpwd2Elt.value;
                    if(userpwd != userpwd2){
                        // 密码不一致
                        pwdErrorSpan.innerText = "密码不一致";
```

```
                    }else{
                        // 密码一致
                    }
                }

                // 绑定focus事件
                userpwd2Elt.onfocus = function(){
                    if(pwdErrorSpan.innerText != ""){
                        userpwd2Elt.value = "";
                    }
                    pwdErrorSpan.innerText = "";
                }

                // 获取email的span
                var emailSpan = document.getElementById("emailError");
                // 给email绑定blur事件
                var emailElt = document.getElementById("email");
                emailElt.onblur = function(){
                    // 获取email
                    var email = emailElt.value;
                    // 编写email的正则
                    var emailRegExp = /^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+
([-.]\w+)*$/;
                    var ok = emailRegExp.test(email);
                    if(ok){
                        // 合法
                    }else{
                        // 不合法
                        emailSpan.innerText = "邮箱地址不合法";
                    }
                }

                // 给emailElt绑定focus
                emailElt.onfocus = function(){
                    if(emailSpan.innerText != ""){
                        emailElt.value = "";
                    }
                    emailSpan.innerText = "";
                }

                // 给提交按钮绑定鼠标单击事件
                var submitBtnElt = document.getElementById("submitBtn");
                submitBtn.onclick = function(){
                    // 触发username的blur userpwd2的blur email的blur
                    // 不需要人工操作,使用纯JS代码触发事件.
                    usernameElt.focus();
                    usernameElt.blur();

                    userpwd2Elt.focus();
                    userpwd2Elt.blur();

                    emailElt.focus();
                    emailElt.blur();

                    // 当所有表单项都是合法的时候,提交表单
                    if(usernameErrorSpan.innerText == "" &&
pwdErrorSpan.innerText == "" && emailSpan.innerText == ""){
                        // 获取表单对象
```

```
                        var userFormElt = document.getElementById("userForm");
                        // 可以在这里设置action,也可以不在这里.
                        userFormElt.action = "http://localhost:8080/jd/save";
                        // 提交表单
                        userFormElt.submit();
                    }
                }
            }
        </script>

        <!--这个表单提交应该使用post，这里为了检测，所以使用get。-->
        <!-- <form id="userForm" action="http://localhost:8080/jd/save"
method="get"> -->
        <form id="userForm" method="get">
            用户名<input type="text" name="username" id="username"/><span
id="usernameError"></span><br>
            密码<input type="text" name="userpwd" id="userpwd"/><br>
            确认密码<input type="text" id="userpwd2" /><span id="pwdError"></span>
<br>
            邮箱<input type="text" name="email" id="email" /><span
id="emailError"></span><br>
            <!-- <input type="submit" value="注册" /> -->
            <input type="button" value="注册" id="submitBtn"/>
            <input type="reset" value="重置" />
        </form>

    </body>
</html>
```

006-复选框的全选和取消全选.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>复选框的全选和取消全选</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
            window.onload = function(){
                var firstChk = document.getElementById("firstChk");
                firstChk.onclick = function(){
                    // 获取第一个复选框的选中状态(复选框对象checkbox对象)
                    //alert(firstChk.checked);
                    // 根据name获取所有元素
                    var aihaos = document.getElementsByName("aihao");
                    if(firstChk.checked){
                        // 全选
                        for(var i = 0; i < aihaos.length; i++){
                            aihaos[i].checked = true;
                        }
                    }else{
                        // 取消全选
                        for(var i = 0; i < aihaos.length; i++){
                            aihaos[i].checked = false;
```

```
                    }
                }
            }
        }
        */

        window.onload = function(){

            var aihaos = document.getElementsByName("aihao");
            var firstChk = document.getElementById("firstChk");
            firstChk.onclick = function(){
                for(var i = 0; i < aihaos.length; i++){
                    aihaos[i].checked = firstChk.checked;
                }
            }

            // 对以上数组进行遍历
            var all = aihaos.length;
            for(var i = 0; i < aihaos.length; i++){
                aihaos[i].onclick = function(){
                    var checkedCount = 0;
                    // 总数量和选中的数量相等的时候,第一个复选框选中.
                    for(var i = 0; i < aihaos.length; i++){
                        if(aihaos[i].checked){
                            checkedCount++;
                        }
                    }
                    firstChk.checked = (all == checkedCount);
                    /*
                    if(all == checkedCount){
                        firstChk.checked = true;
                    }else{
                        firstChk.checked = false;
                    }
                    */
                }
            }
        }
    </script>
    <input type="checkbox" id="firstChk"/><Br>
    <input type="checkbox" name="aihao" value="smoke" />抽烟<Br>
    <input type="checkbox" name="aihao" value="drink" />喝酒<Br>
    <input type="checkbox" name="aihao" value="tt" />烫头<Br>
</body>
</html>
```

007-获取下拉列表选中项的value.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>获取下拉列表选中项的value</title>
    </head>
    <body>
        <!--
```

```
        <select onchange="alert(this.value)">
            <option value="">--请选择省份--</option>
            <option value="001">河北省</option>
            <option value="002">河南省</option>
            <option value="003">山东省</option>
            <option value="004">山西省</option>
        </select>
        -->

        <script type="text/javascript">
            window.onload = function(){
                var provinceListElt = document.getElementById("provinceList");
                provinceListElt.onchange = function(){
                    // 获取选中项的value
                    alert(provinceListElt.value);
                }
            }
        </script>
        <select id="provinceList">
            <option value="">--请选择省份--</option>
            <option value="001">河北省</option>
            <option value="002">河南省</option>
            <option value="003">山东省</option>
            <option value="004">山西省</option>
        </select>

    </body>
</html>

<!--
省份和市区的关系是：1对多
省份表t_province
id      pcode      pname
----------------------------
1       001        河北省
2       002        河南省
3       003        山东省
4       004        山西省

市区表t_city
id      ccode      cname       pcode(fk)
-----------------------------------------------
1       101        石家庄         001
2       102        保定          001
3       103        邢台          001
4       104        承德          001
5       105        张家口          001
6       106        邯郸          001
7       107        衡水          001

前端用户选择的假设是河北省，那么必须获取到河北省的pcode，获取到001
然后将001发送提交给服务器，服务器底层执行一条SQL语句：
    select * from t_city where pcode = '001';

    返回一个List集合，List<City> cityList;

    cityList响应浏览器，浏览器在解析cityList集合转换成一个新的下拉列表。
```

```
-->
```

008-显示网页时钟.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>显示网页时钟</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                关于JS中内置的支持类：Date，可以用来获取时间/日期。
            */
           // 获取系统当前时间
           var nowTime = new Date();
           // 输出
           // document.write(nowTime);
           // 转换成具有本地语言环境的日期格式.
           nowTime = nowTime.toLocaleString();
           document.write(nowTime);
           document.write("<br>");
           document.write("<br>");
           // 当以上格式不是自己想要的,可以通过日期获取年月日等信息,自定制日期格式.
           var t = new Date();
           var year = t.getFullYear(); // 返回年信息,以全格式返回.
           var month = t.getMonth(); // 月份是:0-11
           // var dayOfWeek = t.getDay(); // 获取的一周的第几天(0-6)
           var day = t.getDate(); // 获取日信息.
           document.write(year + "年" + (month+1) + "月" + day + "日");

           document.write("<br>");
           document.write("<br>");

           // 重点:怎么获取毫秒数? (从1970年1月1日 00:00:00 000到当前系统时间的总毫秒数)
           //var times = t.getTime();
           //document.write(times); // 一般会使用毫秒数当做时间戳. (timestamp)

           document.write(new Date().getTime());

        </script>

        <script type="text/javascript">
            function displayTime(){
                var time = new Date();
                var strTime = time.toLocaleString();
                document.getElementById("timeDiv").innerHTML = strTime;
            }

            // 每隔1秒调用displayTime()函数
            function start(){
                // 从这行代码执行结束开始,则会不间断的,每隔1000毫秒调用一次displayTime()
函数.
                v = window.setInterval("displayTime()", 1000);
            }
```

```
            function stop(){
                window.clearInterval(v);
            }
        </script>
        <br><br>
        <input type="button" value="显示系统时间" onclick="start();"/>
        <input type="button" value="系统时间停止" onclick="stop();" />
        <div id="timeDiv"></div>
    </body>
</html>
```

009-内置支持类Array.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>内置支持类Array</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
            // 创建长度为0的数组
            var arr = [];
            alert(arr.length);

            // 数据类型随意
            var arr2 = [1,2,3,false,"abc",3.14];
            alert(arr2.length);

            // 下标会越界吗
            arr2[7] = "test"; // 自动扩容.

            document.write("<br>");

            // 遍历
            for(var i = 0; i < arr2.length; i++){
                document.write(arr2[i] + "<br>");
            }

            // 另一种创建数组的对象的方式
            var a = new Array();
            alert(a.length); // 0

            var a2 = new Array(3); // 3表示长度.
            alert(a2.length);

            var a3 = new Array(3,2);
            alert(a3.length); // 2
            */

           var a = [1,2,3,9];
           var str = a.join("-");
           alert(str); // "1-2-3-9"

           // 在数组的末尾添加一个元素(数组长度+1)
```

```
            a.push(10);
            alert(a.join("-"));

            // 将数组末尾的元素弹出(数组长度-1)
            var endElt = a.pop();
            alert(endElt);
            alert(a.join("-"));

            // 注意:JS中的数组可以自动模拟栈数据结构:后进先出,先进后出原则.
            // push压栈
            // pop弹栈

            // 反转数组.
            a.reverse();
            alert(a.join("="));
        </script>
    </body>
</html>
```

1.4 ECMAScript

004-关于JS中的变量.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>关于JS中的变量</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                回顾java中的变量:
                    1、java中怎么定义/声明变量?
                        数据类型 变量名;
                        例如:
                            int i;
                            double d;
                            boolean flag;
                    2、java中的变量怎么赋值?
                        使用“=”运算符进行赋值运算。("="运算符右边先执行，将右边执行的结果赋值给左边的变量。)
                        变量名 = 值;
                        例如:
                            i = 10;
                            d = 3.14;
                            flag = false;
                    3、java语言是一种强类型语言，强类型怎么理解?
                        java语言存在编译阶段，假设有代码：int i;
                        那么在Java中有一个特点是：java程序编译阶段就已经确定了
                        i变量的数据类型，该i变量的数据类型在编译阶段是int类型，
                        那么这个变量到最终内存释放，一直都是int类型，不可能变成
                        其他类型。
                            int i = 10;
                            double d = i;
```

这行代码是说声明一个新的变量d，double类型，把i变量中保存的值传给d。

i还是int类型。

i = "abc"; 这行代码编译的时候会报错，因为i变量的数据类型是int类型，不能将字符串赋给i。

java中要求变量声明的时候是什么类型，以后永远都是这种类型，不可变。编译期强行固定变量的数据类型。

称为强类型语言。

```
public void sum(int a, int b){}
sum(?,?);
```

javascript当中的变量？

怎么声明变量？

```
var 变量名;
```

怎么给变量赋值？

```
变量名 = 值;
```

javascript是一种弱类型语言，没有编译阶段，一个变量可以随意赋值，赋什么类型的值都行。

```
var i = 100;
i = false;
i = "abc";
i = new Object();
i = 3.14;
```

重点：javascript是一种弱类型编程语言。

```
 */
// 在JS当中,当一个变量没有手动赋值的时候,系统默认赋值undefined
var i;
// undefined 在JS中是一个具体存在值.
alert("i = " + i); // i = undefined

alert(undefined);
var k = undefined;
alert("k = " + k);

// 一个变量没有声明/定义,直接访问?
// alert(age); //语法错误：age is not defined (变量age不存在。不能这样写。)

var a, b, c = 200;
alert("a = " + a);
alert("b = " + b);
alert("c = " + c);

a = false;
alert(a);

a = "abc";
alert(a);

a = 1.2;
alert(a);

    </script>
</body>
```

```
</html>
```

005-JS函数初步.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS函数初步</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、JS中的函数：
                    等同于java语言中的方法，函数也是一段可以被重复利用的代码片段。
                    函数一般都是可以完成某个特定功能的。

                2、回顾java中的方法？
                    [修饰符列表] 返回值类型 方法名(形式参数列表){
                        方法体;
                    }
                    例如：
                    public static boolean login(String username,String password)
{
                        ...
                        return true;
                    }
                    boolean loginSuccess = login("admin","123");

                3、JS中的变量是一种弱类型的，那么函数应该怎么定义呢？
                    语法格式：
                        第一种方式：
                            function 函数名(形式参数列表){
                                函数体;
                            }
                        第二种方式：
                            函数名 = function(形式参数列表){
                                函数体;
                            }

                        JS中的函数不需要指定返回值类型，返回什么类型都行。
             */
            function sum(a, b){
                // a和b都是局部变量,他们都是形参(a和b都是变量名，变量名随意。)
                alert(a + b);
            }

            // 函数必须调用才能执行的.
            //sum(10, 20);

            // 定义函数sayHello
            sayHello = function(username){
                alert("hello " + username);
            }

            // 调用函数
```

```
            //sayHello("zhangsan");

        </script>

        <input type="button" value="hello" onclick="sayHello('jack');" />
        <input type="button" value="计算10和20的求和" onclick="sum(10, 20);" />

    </body>
</html>
```

006-JS函数初步.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS函数初步</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                java中的方法有重载机制，JS中的函数能重载吗？
                    JS当中的函数在调用的时候，参数的类型没有限制，并且参数的个数也没有限制，
JS就是这么随意。（弱类型）

                重载的含义：
                    方法名或者函数名一样，形参不同（个数、类型、顺序）
             */
            function sum(a, b){
                return a + b;
            }

            // 调用函数sum
            var retValue = sum(1, 2);
            alert(retValue);

            var retValue2 = sum("jack"); // jack赋值给a变量,b变量没有赋值系统默认赋值
undefined
            alert(retValue2); // jackundefined

            var retValue3 = sum();
            alert(retValue3); // NaN (NaN是一个具体存在的值，该值表示不是数字。Not a
Number)

            var retValue4 = sum(1, 2, 3);
            alert("结果=" + retValue4); // 结果=3

            function test1(username){
                alert("test1");
            }

            /*
            在JS当中，函数的名字不能重名，当函数重名的时候，后声明的函数会将之前声明的同名函数
覆盖。
            */
            function test1(){
```

```
            alert("test1 test1");
        }

        test1("lisi"); // 这个调用的是第二个test1()函数.

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

007-JS的局部变量和全局变量.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS的局部变量和全局变量</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                全局变量：
                    在函数体之外声明的变量属于全局变量，全局变量的生命周期是：
                        浏览器打开时声明，浏览器关闭时销毁，尽量少用。因为全局变量会一直在浏览器的内存当中，耗费内存空间。
                        能使用局部变量尽量使用局部变量。
                局部变量：
                    在函数体当中声明的变量，包括一个函数的形参都属于局部变量，
                    局部变量的生命周期是：函数开始执行时局部变量的内存空间开辟，函数执行结束之后，局部变量的内存空间释放。
                    局部变量生命周期较短。
             */

            // 全局变量
            var i = 100;

            function accessI(){
                // 访问的是全局变量
                alert("i = " + i);
            }

            accessI();

            // 全局变量
            var username = "jack";
            function accessUsername(){
                // 局部变量
                var username = "lisi";
                // 就近原则:访问局部变量
                alert("username = " + username);
            }
            // 调用函数
            accessUsername();
            // 访问全局变量
            alert("username = " + username);

            function accessAge(){
```

```
                var age = 20;
                alert("年龄 = " + age);
            }

            accessAge();

            // 报错(语法不对)
            // alert("age = " + age);

            // 以下语法是很奇怪的.
            function myfun(){
                // 当一个变量声明的时候没有使用var关键字,那么不管这个变量是在哪里声明的,都是全局变量.
                myname = "dujubin";
            }

            // 访问函数
            myfun();

            alert("myname = " + myname); // myname = dujubin

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

008-JS中的数据类型.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS中的数据类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、虽然JS中的变量在声明的时候不需要指定数据类型，但是在赋值，每一个数据还是有类型的，所以
                这里也需要学习一下JS包括哪些数据类型？
                    JS中数据类型有：原始类型、引用类型。
                    原始类型：Undefined、Number、String、Boolean、Null
                    引用类型：Object以及Object的子类

                2、ES规范(ECMAScript规范)，在ES6之后，又基于以上的6种类型之外添加了一种新的类型：Symbol

                3、JS中有一个运算符叫做typeof，这个运算符可以在程序的运行阶段动态的获取变量的数据类型。
                    typeof运算符的语法格式：
                        typeof 变量名
                    typeof运算符的运算结果是以下6个字符串之一：注意字符串都是全部小写。
                        "undefined"
                        "number"
                        "string"
                        "boolean"
                        "object"
```

```
                        "function"

                4、在JS当中比较字符串是否相等使用“==”完成。没有equals。
             */

            /*
            // 求和,要求a变量和b变量将来的数据类型必须是数字,不能是其他类型
            // 因为以下定义的这个sum函数是为了完成两个数字的求和.
            function sum(a, b){
                if(typeof a == "number" && typeof b == "number"){
                    return a + b;
                }
                alert(a + "," + b + "必须都为数字! ");
            }

            // 别人去调用以上你写的sum函数.
            var retValue = sum(false, "abc");
            alert(retValue); // undefined

            var retValue2 = sum(1, 2);
            alert(retValue2); // 3
            */

           var i;
           alert(typeof i); // "undefined"

           var k = 10;
           alert(typeof k); // "number"

           var f = "abc";
           alert(typeof f); // "string"

           var d = null;
           alert(typeof d); // "object"  null属于Null类型,但是typeof运算符的结果
是"object"

           var flag = false;
           alert(typeof flag); // "boolean"

           var obj = new Object();
           alert(typeof obj); // "object"

           // sayHello是一个函数.
           function sayHello(){

           }
           alert(typeof sayHello); // "function"


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

009-Undefined类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Undefined类型</title>
    </head>
    <body>

        <script type="text/javascript">
            /*
                Undefined类型只有一个值，这个值就是 undefined
                当一个变量没有手动赋值，系统默认赋值undefined
                或者也可以给一个变量手动赋值undefined。
            */
           var i; // undefined
           var k = undefined; // undefined

           alert(i == k); // true

           var y = "undefined"; // "undefined"
           alert(y == k); // false

        </script>

    </body>
</html>
```

010-Number类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Number类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、Number类型包括哪些值？
                    -1 0 1 2 2.3 3.14 100 .... NaN Infinity
                    整数、小数、正数、负数、不是数字、无穷大都属于Number类型。
                2、isNaN() : 结果是true表示不是一个数字，结果是false表示是一个数字。
                3、parseInt()函数
                4、parseFloat()函数
                5、Math.ceil() 函数（Math是数学类，数学类当中有一个函数叫做ceil()，作用是
向上取整。）
            */
           var v1 = 1;
           var v2 = 3.14;
           var v3 = -100;
           var v4 = NaN;
           var v5 = Infinity;
```

```
            // "number"
            alert(typeof v1);
            alert(typeof v2);
            alert(typeof v3);
            alert(typeof v4);
            alert(typeof v5);

            // 关于NaN (表示Not a Number，不是一个数字，但属于Number类型)
            // 什么情况下结果是一个NaN呢?
            // 运算结果本来应该是一个数字,最后算完不是一个数字的时候,结果是NaN.
            var a = 100;
            var b = "中国人";
            alert(a / b); // 除号显然最后结果应该是一个数字,但是运算的过程中导致最后不是一个数字,那么最后的结果是NaN

            var e = "abc";
            var f = 10;
            alert(e + f); // "abc10"

            // Infinity (当除数为0的时候，结果为无穷大)
            alert(10 / 0);

            // 思考:在JS中10 / 3 = ?
            alert(10 / 3); // 3.3333333333333335

            // 关于isNaN函数?
            // 用法：isNaN(数据) ,结果是true表示不是一个数字, 结果是false表示是一个数字.
            // isNaN : is Not a Number
            function sum(a, b){
                if(isNaN(a) || isNaN(b)){
                    alert("参与运算的必须是数字! ");
                    return;
                }
                return a + b;
            }
            sum(100, "abc");
            alert(sum(100, 200));

            // parseInt():可以将字符串自动转换成数字,并且取整数位.
            alert(parseInt("3.9999")); // 3
            alert(parseInt(3.9999)); // 3

            // parseFloat():可以将字符串自动转换成数字.
            alert(parseFloat("3.14") + 1); // 4.14
            alert(parseFloat("3.2") + 1); // 4.2

            // Math.ceil()
            alert(Math.ceil("2.1")); // 3

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

011-Boolean类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Boolean类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
            1、 JS中的布尔类型永远都只有两个值：true和false （这一点和java相同。）
            2、在Boolean类型中有一个函数叫做：Boolean()。
                语法格式：
                    Boolean(数据)
                Boolean()函数的作用是将非布尔类型转换成布尔类型。
            */
           // var username = "lucy";
           var username = "";

           /*
           if(Boolean(username)){
               alert("欢迎你" + username);
           }else{
               alert("用户名不能为空！");
           }
           */

          /*
           if(username){
               alert("欢迎你" + username);
           }else{
               alert("用户名不能为空！");
           }
           */

          // 规律:“有"就转换成true,"没有"就转换成false.
          alert(Boolean(1)); // true
          alert(Boolean(0)); // false
          alert(Boolean("")); // false
          alert(Boolean("abc")); // true
          alert(Boolean(null)); // false
          alert(Boolean(NaN)); // false
          alert(Boolean(undefined)); // false
          alert(Boolean(Infinity)); // true

           /*
           while(10 / 3){
               alert("hehe");
           }
           */

           for(var i = 0; i < 10; i++){
               alert("i = " + i);
           }

           // Null类型只有一个值,null
```

```
            alert(typeof null); // "object"

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

012-String类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>String类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                String类型：
                    1、在JS当中字符串可以使用单引号，也可以使用双引号。
                        var s1 = 'abcdef';
                        var s2 = "test";
                    2、在JS当中，怎么创建字符串对象呢？
                        两种方式：
                            第一种：var s = "abc";
                            第二种（使用JS内置的支持类String）： var s2 = new
String("abc");
                        需要注意的是：String是一个内置的类，可以直接用，String的父类是
Object。
                    3、无论小string还是大String，他们的属性和函数都是通用的。
                    4、关于String类型的常用属性和函数？
                        常用属性：
                            length 获取字符串长度
                        常用函数：
                            indexOf          获取指定字符串在当前字符串中第一次出现处的索
引
                            lastIndexOf      获取指定字符串在当前字符串中最后一次出现处的
索引
                            replace          替换
                            substr           截取子字符串
                            substring        截取子字符串
                            toLowerCase      转换小写
                            toUpperCase      转换大写
                            split            拆分字符串
            */
            // 小string(属于原始类型String)
            var x = "king";
            alert(typeof x); // "string"

            // 大String(属于Object类型)
            var y = new String("abc");
            alert(typeof y); // "object"

            // 获取字符串的长度
            alert(x.length); // 4
            alert(y.length); // 3
```

```
            alert("http://www.baidu.com".indexOf("http")); // 0
            alert("http://www.baidu.com".indexOf("https")); // -1

            // 判断一个字符串中是否包含某个子字符串？
            alert("http://www.baidu.com".indexOf("https") >= 0 ? "包含" : "不包
含"); // 不包含

            // replace (注意：只替换了第一个)
            alert("name=value%name=value%name=value".replace("%","&")); //
name=value&name=value%name=value

            // 继续调用replace方法,就会替换第“二”个.
            // 想全部替换需要使用正则表达式.

alert("name=value%name=value%name=value".replace("%","&").replace("%", "&")); //
name=value&name=value&name=value

            // 考点:经常问 substr和substring的区别？
            // substr(startIndex, length)
            alert("abcdefxyz".substr(2,4)); //cdef
            // substring(startIndex, endIndex) 注意:不包含endIndex
            alert("abcdefxyz".substring(2,4)); //cd

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

013-Object类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Object类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                Object类型：
                    1、Object类型是所有类型的超类，自定义的任何类型，默认继承Object。
                    2、Object类包括哪些属性？
                        prototype属性（常用的，主要是这个）：作用是给类动态的扩展属性和函
数。
                        constructor属性
                    3、Object类包括哪些函数？
                        toString()
                        valueOf()
                        toLocaleString()
                    4、在JS当中定义的类默认继承Object，会继承Object类中所有的属性以及函
数。
                    换句话说，自己定义的类中也有prototype属性。

                    5、在JS当中怎么定义类？怎么new对象？
                        定义类的语法：
                            第一种方式：
                                function 类名(形参){
```

```
                        }
                    第二种方式：
                        类名 = function(形参){

                        }
                创建对象的语法：
                    new 构造方法名(实参); // 构造方法名和类名一致。
 */
function sayHello(){

}

// 把sayHello当做一个普通的函数来调用.
sayHello();

// 这种方式就表示把sayHello当做一个类来创建对象.
var obj = new sayHello(); // obj是一个引用,保存内存地址指向堆中的对象.

// 定义一个学生类
function Student(){
    alert("Student.....");
}

// 当做普通函数调用
Student();

// 当做类来创建对象
var stu = new Student();
alert(stu); // [object Object]

// JS中的类的定义,同时又是一个构造函数的定义
// 在JS中类的定义和构造函数的定义是放在一起来完成的.
function User(a, b, c){ // a b c是形参,属于局部变量.
    // 声明属性 (this表示当前对象)
    // User类中有三个属性:sno/sname/sage
    this.sno = a;
    this.sname = b;
    this.sage = c;
}

// 创建对象
var u1 = new User(111, "zhangsan", 30);
// 访问对象的属性
alert(u1.sno);
alert(u1.sname);
alert(u1.sage);

var u2 = new User(222, "jackson", 55);
alert(u2.sno);
alert(u2.sname);
alert(u2.sage);

// 访问一个对象的属性,还可以使用这种语法
alert(u2["sno"]);
alert(u2["sname"]);
alert(u2["sage"]);
```

```
            // 定义类的另一种语法
            /*
            Emp = function(a, b){
                this.ename = a;
                this.sal = b;
            }
            */

            Emp = function(ename,sal){
                // 属性
                this.ename = ename;
                this.sal = sal;
            }

            var e1 = new Emp("SMITH", 800);
            alert(e1["ename"] + "," + e1.sal);

            Product = function(pno,pname,price){
                // 属性
                this.pno = pno;
                this.pname = pname;
                this.price = price;
                // 函数
                this.getPrice = function(){
                    return this.price;
                }
            }

            var xigua = new Product(111, "西瓜", 4.0);
            var pri = xigua.getPrice();
            alert(pri); // 4.0

            // 可以通过prototype这个属性来给类动态扩展属性以及函数
            Product.prototype.getPname = function(){
                return this.pname;
            }

            // 调用后期扩展的getPname()函数
            var pname = xigua.getPname();
            alert(pname)

            // 给String扩展一个函数
            String.prototype.suiyi = function(){
                alert("这是给String类型扩展的一个函数，叫做suiyi");
            }

            "abc".suiyi();

        </script>
    </body>
</html>
<!--
    java语言怎么定义类，怎么创建对象？（强类型）
        public class User{
            private String username;
            private String password;
            public User(){
```

```
            }
            public User(String username,String password){
                this.username = username;
                this.password = password;
            }
        }
        User user = new User();
        User user = new User("lisi","123");

    JS语言怎么定义类，怎么创建对象？（弱类型）
        User = function(username,password){
            this.username = username;
            this.password = password;
        }
        var u = new User();
        var u = new User("zhangsan");
        var u = new User("zhangsan","123");
-->
```

014-null NaN undefined这三个值有什么区别.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>null NaN undefined这三个值有什么区别</title>
    </head>
    <body>
        <script type="text/javascript">
            // == 是等同运算符
            alert(1 == true); // true
            alert(1 === true); // false

            // null NaN undefined 数据类型不一致.
            alert(typeof null); // "object"
            alert(typeof NaN); // "number"
            alert(typeof undefined); // "undefined"

            // null和undefined可以等同.
            alert(null == NaN); // false
            alert(null == undefined); // true
            alert(undefined == NaN); // false

            // 在JS当中有两个比较特殊的运算符
            // ==(等同运算符：只判断值是否相等)
            // ===(全等运算符：既判断值是否相等，又判断数据类型是否相等)
            alert(null === NaN); // false
            alert(null === undefined); // false
            alert(undefined === NaN); // false
        </script>
    </body>
</html>
```

015-JS的常用事件-注册事件的两种方式.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS的常用事件</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                JS中的事件：

                    blur失去焦点
                    focus获得焦点

                    click鼠标单击
                    dblclick鼠标双击

                    keydown键盘按下
                    keyup键盘弹起

                    mousedown鼠标按下
                    mouseover鼠标经过
                    mousemove鼠标移动
                    mouseout鼠标离开
                    mouseup鼠标弹起

                    reset表单重置
                    submit表单提交

                    change下拉列表选中项改变，或文本框内容改变
                    select文本被选定
                    load页面加载完毕（整个HTML页面中所有的元素全部加载完毕之后发生。）

                任何一个事件都会对应一个事件句柄，事件句柄是在事件前添加on。
                onXXX这个事件句柄出现在一个标签的属性位置上。（事件句柄以属性的形式存在。）
             */
            // 对于当前程序来说,sayHello函数被称为回调函数(callback函数)
            // 回调函数的特点:自己把这个函数代码写出来了,但是这个函数不是自己负责调用,由其他程序负责调用该函数.
            function sayHello(){
                alert("hello js!");
            }
        </script>

        <!--注册事件的第一种方式，直接在标签中使用事件句柄-->
        <!--以下代码的含义是：将sayHello函数注册到按钮上，等待click事件发生之后，该函数被浏览器调用。我们称这个函数为回调函数。-->
        <input type="button" value="hello" onclick="sayHello()"/>


        <input type="button" value="hello2" id="mybtn" />
        <input type="button" value="hello3" id="mybtn1" />
        <input type="button" value="hello4" id="mybtn2" />
        <script type="text/javascript">
            function doSome(){
```

```
                alert("do some!");
            }
            /*
                第二种注册事件的方式，是使用纯JS代码完成事件的注册。
            */
            // 第一步:先获取这个按钮对象(document是全部小写，内置对象，可以直接用，document就代表整个HTML页面)
            var btnObj = document.getElementById("mybtn");
            // 第二步:给按钮对象的onclick属性赋值
            btnObj.onclick = doSome; // 注意:千万别加小括号. btnObj.onclick = doSome();这是错误的写法.
                                     // 这行代码的含义是,将回调函数doSome注册到click事件上.

            var mybtn1 = document.getElementById("mybtn1");
            mybtn1.onclick = function(){ // 这个函数没有名字,叫做匿名函数,这个匿名函数也是一个回调函数.
                alert("test.........."); // 这个函数在页面打开的时候只是注册上,不会被调用,在click事件发生之后才会调用.
            }

            document.getElementById("mybtn2").onclick = function(){
                alert("test22222222.........");
            }
        </script>

    </body>
</html>

<!--
    java中也有回调函数机制:
        public class MyClass{

            public static void main(String[] args){
                // 主动调用run()方法，站在这个角度看run()方法叫做正向调用。
                run();
            }

            // 站在run方法的编写者角度来看这个方法，把run方法叫做回调函数。
            public static void run(){
                System.out.println("run...");
            }
        }
-->
```

016-关于JS代码的执行顺序.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS代码的执行顺序</title>
    </head>
    <!-- load事件什么时候发生？页面全部元素加载完毕之后才会发生。-->
    <body onload="ready()">
```

```
        <script type="text/javascript">

            /*
            // 第一步:根据id获取节点对象
            var btn = document.getElementById("btn"); // 返回null(因为代码执行到此处的时候id="btn"的元素还没有加载到内存)

            // 第二步:给节点对象绑定事件
            btn.onclick = function(){
                alert("hello js");
            }
            */

           function ready(){
               var btn = document.getElementById("btn");
               btn.onclick = function(){
                alert("hello js");
               }
           }

        </script>

        <input type="button" value="hello" id="btn" />

    </body>
</html>
```

017-JS代码的执行顺序.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS代码的执行顺序</title>
    </head>
    <body>

        <script type="text/javascript">
            // 页面加载的过程中,将a函数注册给了load事件
            // 页面加载完毕之后,load事件发生了,此时执行回调函数a
            // 回调函数a执行的过程中,把b函数注册给了id="btn"的click事件
            // 当id="btn"的节点发生click事件之后,b函数被调用并执行.
            window.onload = function(){ // 这个回调函数叫做a
                document.getElementById("btn").onclick = function(){ // 这个回调函数叫做b
                    alert("hello js.........");
                }
            }

        </script>

        <input type="button" value="hello" id="btn" />

    </body>
</html>
```

018-JS代码设置节点的属性.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS代码设置节点的属性</title>
    </head>
    <body>

        <script type="text/javascript">
            window.onload = function(){
                document.getElementById("btn").onclick = function(){
                    var mytext = document.getElementById("mytext");
                    // 一个节点对象中只要有的属性都可以"."
                    mytext.type = "checkbox";
                }
            }
        </script>

        <input type="text" id="mytext"/>

        <input type="button" value="将文本框修改为复选框" id="btn"/>

    </body>
</html>
```

019-JS代码捕捉回车键.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS代码捕捉回车键</title>
    </head>
    <body>
        <script type="text/javascript">
            window.onload = function(){
                var usernameElt = document.getElementById("username");
                // 回车键的键值是13
                // ESC键的键值是27
                /*
                usernameElt.onkeydown = function(a, b, c){
                    // 获取键值
                    // alert(a); // [object KeyboardEvent]
                    // alert(b);
                    // alert(c);
                }
                */
               /*
               usernameElt.onkeydown = function(event){
                // 获取键值
                // 对于“键盘事件对象"来说,都有keyCode属性用来获取键值.
                alert(event.keyCode);
```

```
                }
                */
               usernameElt.onkeydown = function(event){
                   if(event.keyCode === 13){
                       alert("正在进行验证....");
                   }
               }

             }

             // 张三程序员写了这样一个函数
             /*
             function sum(){
                 alert("sum execute......!");
             }
             */

            /*
            function sum(a){
             // alert("sum execute......!");
             alert(a);
            }

             // 李四程序员调用sum函数
             // sum();
             sum("hello world!");
             */
        </script>

        <input type="text" id="username"/>

    </body>
</html>
```

020-JS的void运算符.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS的void运算符</title>
    </head>
    <body>
        页面顶部<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
```

```
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
```

```
		<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
		<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
		<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
		<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
		<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>

		<!--
			void运算符的语法：void(表达式)
			运算原理：执行表达式，但不返回任何结果。
				javascript:void(0)
				其中javascript:作用是告诉浏览器后面是一段JS代码。
				以下程序的javascript:是不能省略的。
		-->
		<a href="javascript:void(0)" onclick="window.alert('test code')">
			既保留住超链接的样式，同时用户点击该超链接的时候执行一段JS代码，但页面还不能跳转。
		</a>

		<br>

		<a href="javascript:void(100)" onclick="window.alert('test code')">
			既保留住超链接的样式，同时用户点击该超链接的时候执行一段JS代码，但页面还不能跳转。
		</a>

		<br>

		<!--void() 这个小括号当中必须有表达式-->
		<!--
		<a href="javascript:void()" onclick="window.alert('test code')">
			既保留住超链接的样式，同时用户点击该超链接的时候执行一段JS代码，但页面还不能跳转。
		</a>
		-->

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

021-JS的控制语句.html：

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>JS的控制语句</title>
	</head>
	<body>
		<script type="text/javascript">
			/*
				1、if
				2、switch

				3、while
```

```
                4、do .. while..
                5、for循环

                6、break
                7、continue

                8、for..in语句（了解）
                9、with语句（了解）
             */
            // 创建JS数组
            var arr = [false,true,1,2,"abc",3.14]; // JS中数组中元素的类型随意.元素的个数随意.
            // 遍历数组
            for(var i = 0; i < arr.length; i++){
                alert(arr[i]);
            }

            // for..in
            for(var i in arr){
                //alert(i);
                alert(arr[i]);
            }

            // for..in语句可以遍历对象的属性
            User = function(username,password){
                this.username = username;
                this.password = password;
            }

            var u = new User("张三", "444");
            alert(u.username + "," + u.password);
            alert(u["username"] + "," + u["password"]);

            for(var shuXingMing in u){
                //alert(shuXingMing)
                //alert(typeof shuXingMing) // shuXingMing是一个字符串
                alert(u[shuXingMing]);
            }

            alert(u.username);
            alert(u.password);

            with(u){
                alert(username + "," + password);
            }

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

<!--
    public class Test{
        public static void main(String[] args){
            int[] arr = {1,2,3,4,5,6};
            int[] arr2 = new int[5]; // 等同于：int[] arr2 = {0,0,0,0,0};
            String[] arr3 = {"a","b","c"};
            String[] arr4 = new String[3]; // 等同于：String[] arr4 = {null,null,null};
```

```
        }
    }
-->
```

1.5 JSON

001.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JSON</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、什么是JSON，有什么用？
                    JavaScript Object Notation（JavaScript对象标记），简称JSON。(数据交换格式)
                    JSON主要的作用是：一种标准的数据交换格式。（目前非常流行，90%以上的系统，系统A与系统B交换数据的话，都是采用JSON。）
                2、JSON是一种标准的轻量级的数据交换格式。特点是：
                    体积小，易解析。
                3、在实际的开发中有两种数据交换格式，使用最多，其一是JSON，另一个是XML。
                    XML体积较大，解析麻烦，但是有其优点是：语法严谨。（通常银行相关的系统之间进行数据交换的话会使用XML。）
                4、JSON的语法格式：
                    var jsonObj = {
                        "属性名" : 属性值,
                        "属性名" : 属性值,
                        "属性名" : 属性值,
                        "属性名" : 属性值,
                        ....
                    };
            */
            // 创建JSON对象(JSON也可以称为无类型对象。轻量级，轻巧。体积小。易解析。)
            var studentObj = {
                "sno" : "110",
                "sname" : "张三",
                "sex" : "男"
            };

            // 访问JSON对象的属性
            alert(studentObj.sno + "," + studentObj.sname + "," + studentObj.sex);


            // 之前没有使用JSON的时候,定义类,创建对象,访问对象的属性.
            Student = function(sno,sname,sex){
                this.sno = sno;
                this.sname = sname;
                this.sex = sex;
            }
            var stu = new Student("111","李四","男");
```

```
            alert(stu.sno + "," + stu.sname + "," + stu.sex);

            // JSON数组
            var students = [
                {"sno":"110","sname":"张三","sex":"男"},
                {"sno":"120","sname":"李四","sex":"男"},
                {"sno":"130","sname":"王五","sex":"男"}
            ];

            // 遍历
            for(var i = 0; i < students.length; i++){
                var stuObj = students[i];
                alert(stuObj.sno + "," + stuObj.sname + "," + stuObj.sex);
            }
        </script>
    </body>
</html>
```

002.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>复杂一些的JSON对象。</title>
    </head>
    <body>
        <script type="text/javascript">
            var user = {
                "usercode" : 110,
                "username" : "张三",
                "sex" : true,
                "address" : {
                    "city" : "北京",
                    "street" : "大兴区",
                    "zipcode" : "12212121",
                },
                "aihao" : ["smoke","drink","tt"]
            };

            // 访问人名以及居住的城市
            alert(user.username + "居住在" + user.address.city);

            /*
                请自行设计JSON格式的数据，这个JSON格式的数据可以描述整个班级中每一个学生的信息，以及总人数信息。
            */
            var jsonData = {
                "total" : 3,
                "students" : [
                    {"name":"zhangsan","birth":"1980-10-20"},
                    {"name":"lisi","birth":"1981-10-20"},
                    {"name":"wangwu","birth":"1982-10-20"}
                ]
            };
```

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

003-eval函数.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>eval函数</title>
    </head>
    <body>
        <!--
            JSON是一种行业内的数据交换格式标准。
            JSON在JS中以JS对象的形式存在。
        -->
        <script type="text/javascript">
            /*
                eval函数的作用是：
                    将字符串当做一段JS代码解释并执行。
            */
           /*
           window.eval("var i = 100;");
           alert("i = " + i); // i = 100
           */

           // java连接数据库,查询数据之后,将数据在java程序中拼接成JSON格式的“字符串”,将json格式的字符串响应到浏览器
           // 也就是说java响应到浏览器上的仅仅是一个"JSON格式的字符串",还不是一个json对象.
           // 可以使用eval函数,将json格式的字符串转换成json对象.
           var fromJava = "{\"name\":\"zhangsan\",\"password\":\"123\"}"; //这是java程序给发过来的json格式的"字符串"
           // 将以上的json格式的字符串转换成json对象
           window.eval("var jsonObj = " + fromJava);
           // 访问json对象
           alert(jsonObj.name + "," + jsonObj.password); // 在前端取数据.

           /*
            面试题：
                在JS当中：[]和{}有什么区别？
                    [] 是数组。
                    {} 是JSON。

                java中的数组：int[] arr = {1,2,3,4,5};
                JS中的数组：var arr = [1,2,3,4,5];
                JSON：var jsonObj = {"email" : "zhangsan@123.com","age":25};
           */

           var json = {
               "username" : "zhangsan"
           };
           // JS中访问json对象的属性
           alert(json.username);
```

```
            // JS中访问json对象的属性
            alert(json["username"]);

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

004-设置table的tbody.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>设置table的tbody</title>
    </head>
    <body>
        <script type="text/javascript">
            // 有这些json数据
            var data = {
                "total" : 4,
                "emps" : [
                    {"empno":7369,"ename":"SMITH","sal":800.0},
                    {"empno":7361,"ename":"SMITH2","sal":1800.0},
                    {"empno":7360,"ename":"SMITH3","sal":2800.0},
                    {"empno":7362,"ename":"SMITH4","sal":3800.0}
                ]
            };

            // 希望把数据展示到table当中.
            window.onload = function(){
                var displayBtnElt = document.getElementById("displayBtn");
                displayBtnElt.onclick = function(){
                    var emps = data.emps;
                    var html = "";
                    for(var i = 0; i < emps.length; i++){
                        var emp = emps[i];
                        html += "<tr>";
                        html += "<td>"+emp.empno+"</td>";
                        html += "<td>"+emp.ename+"</td>";
                        html += "<td>"+emp.sal+"</td>";
                        html += "</tr>";
                    }
                    document.getElementById("emptbody").innerHTML = html;
                    document.getElementById("count").innerHTML = data.total;
                }
            }
        </script>
        <input type="button" value="显示员工信息列表" id="displayBtn" />
        <h2>员工信息列表</h2>
        <hr>
        <table border="1px" width="50%">
            <tr>
                <th>员工编号</th>
                <th>员工名字</th>
                <th>员工薪资</th>
            </tr>
```

```
            <tbody id="emptbody">
                <!--
                <tr>
                    <td>7369</td>
                    <td>SMITH</td>
                    <td>800</td>
                </tr>
                <tr>
                    <td>7369</td>
                    <td>SMITH</td>
                    <td>800</td>
                </tr>
                <tr>
                    <td>7369</td>
                    <td>SMITH</td>
                    <td>800</td>
                </tr>
                -->
            </tbody>
        </table>
        总共<span id="count">0</span>条数
    </body>
</html>
```