javascript

1.1 BOM

001-窗口的开启和关闭.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>窗口的开启和关闭</title>
    </head>
    <body>
        <!-- window.open开启一个新窗口 -->
        <!-- window.open(url, target) ，和超链接效果差不多，都是可以发送请求给服务器的-->

        <!-- 开启一个窗口，默认是开启一个新窗口。 -->
        <input type="button" value="开启百度"
onclick="window.open('http://www.baidu.com')"/>

        <!-- 开启一个窗口，在当前窗口中显示 -->
        <input type="button" value="开启百度"
onclick="window.open('http://www.baidu.com', '_self')"/>
        <!-- 开启一个窗口，在新窗口中显示 -->
        <input type="button" value="开启百度"
onclick="window.open('http://www.baidu.com', '_blank')"/>
        <!-- 开启一个窗口，在父窗口中显示 -->
        <input type="button" value="开启百度"
onclick="window.open('http://www.baidu.com', '_parent')"/>
        <!-- 开启一个窗口，在顶级窗口中显示 -->
        <input type="button" value="开启百度"
onclick="window.open('http://www.baidu.com', '_top')"/>

        <input type="button" value="002" onclick="window.open('002.html')"/>

    </body>
</html>
```

002.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body>
        002窗口
        <input type="button" value="关闭窗口" onclick="window.close()"/>
    </body>
</html>
```

003-alert和confirm方法.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>alert和confirm方法</title>
    </head>
    <body>
        <script type="text/javascript">
            function sayHello(){
                // 弹出消息框
                window.alert("hello world!");
            }

            function del(){
                // 删除数据之前一定要提示用户是否真的删除,用户点击了确定才表示真的删除.
                // 确认框
                //var ok = window.confirm("亲，确认删除数据吗？")
                //console.log(ok) //返回值是一个布尔类型.

                /* if(ok){
                    alert("数据正在删除中，请稍后...")
                } */

                if(window.confirm("亲，确认删除数据吗？")){
                    alert("数据正在删除中，请稍后...")
                }
            }
        </script>

        <input type="button" value="hello" onclick="sayHello()"/>

        <input type="button" value="删除" onclick="del()"/>
    </body>
</html>
```

004-将当前窗口设置为顶级窗口.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>将当前窗口设置为顶级窗口</title>
    </head>
    <body>
        <iframe src="005.html" width="500px" height="500px"></iframe>
    </body>
</html>
```

005.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body>
        005页面
        <script type="text/javascript">
            /* 如果当前这个窗口不是顶级窗口的话，将当前窗口设置为顶级窗口。 */
            function setTop(){
                // window是当前浏览器窗口,代表005.html
                // “当前窗口的顶级窗口"如果"不是自己"
                // window.top就是当前窗口对应的顶级窗口.
                // window.self表示当前自己这个窗口
                // window.top 是004窗口
                // window.self 是005窗口
                //console.log((window.top != window.self))

                if(window.top != window.self){
                    // 将当前窗口设置为顶级窗口
                    // window.self.location 是005的地址
                    // 将顶级窗口的window.top.location地址设置为005
                    window.top.location = window.self.location;
                }
            }
        </script>

        <input type="button" onclick="setTop()" value="如果当前窗口不是顶级窗口的话,将当前窗口设置为顶级窗口"/>

    </body>
</html>
```

006-历史记录.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>历史记录</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></title>
    </head>
    <body>
        007页面007页面007页面007页面007页面007页面007页面007页面007页面007页面007页面
        <!-- back()是后退一步！ -->
        <input type="button" value="后退" onclick="window.history.back()" />
        <!-- go(-1)也是后退一步 -->
        <input type="button" value="后退" onclick="window.history.go(-1)" />
    </body>
</html>
```

008-window.location.href.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>window.location.href</title>
    </head>
    <body>
        <script type="text/javascript">
            function goBaidu(){
                //window.location.href = "http://www.baidu.com";
                //window.location = "http://www.jd.com";

                //document.location.href = "http://www.126.com";
                document.location = "http://www.baidu.com";
            }
        </script>

        <!--
            跳转页面可以通过多种方式：（这些都是发送请求！！！！）
```

```
                第一种方式：直接在浏览器地址栏上写URL。（重点）

                第二种方式：可以点击超链接（重点）

                第三种方式：提交表单（重点）

                第四种方式：window.open(url,target)   （了解）

                第五种方式：js代码（重点）
                            window.location.href
                            window.location
                            document.location.href
                            document.location

                通过浏览器向服务器发送请求，通常是以上的五种方式。
         -->
        <input type="button" value="百度" onclick="goBaidu();"/>
    </body>
</html>
```

1.2 DOM

001-操作div和span.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>操作div和span：innerHTML和innerText属性</title>
        <style type="text/css">
            #div1 {
                background-color: burlywood;
                border: 1px solid red;
                width: 400px;
                height: 50px;
            }

            #span1 {
                background-color: red;
            }
        </style>
    </head>
    <body>

        <script type="text/javascript">
            window.onload = function(){
                document.getElementById("btn").onclick = function(){
                    // 设置div中的内容
                    var divElt = document.getElementById("div1");
                    // 通过元素的innerHTML属性来设置内部的内容
                    // innerHTML 是属性,不是一个方法.
                    // innerHTML属性会将后面的字符串当做一段HTML代码解释并执行!
                    divElt.innerHTML = "<font color='red'>用户名不能为空! </font>";

                    // innerText也可以设置元素当中的内容.和innerHTML有什么区别呢?
```

```
                    // innerText后面的字符串即使是一个HTML代码,也不会当做HTML执行,只是看做普通文本.
                    //divElt.innerText = "<font color='red'>用户名不能为空!</font>";
                }

                var spanbtn = document.getElementById("spanbtn");
                spanbtn.onclick = function(){
                    var span1 = document.getElementById("span1");
                    //span1.innerHTML = "<a href='http://www.baidu.com'>百度</a>";
                    span1.innerText = "<a href='http://www.baidu.com'>百度</a>";
                }

            }
        </script>

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

        <!-- div独占一行！ -->
        <div id="div1"></div>

        <!-- span的大小会随着span中的内容多少变化而变化。 -->
        <span id="span1"></span>

    </body>
</html>
```

002-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">
            // 页面加载完毕之后
            window.onload = function(){

                /* // 给id = "firstChk"元素绑定click
                var firstChkElt = document.getElementById("firstChk");
                firstChkElt.onclick = function(){
                    // 获取到所有的复选框对象
                    var aihaos = document.getElementsByName("aihao");

                    if(firstChkElt.checked){ //get复选框的选中状态
                        // 遍历数组
                        for(var i = 0; i < aihaos.length; i++){
                            var aihaoChk = aihaos[i];
                            aihaoChk.checked = true; //set复选框的选中状态
                        }
                    }else{
```

```
                        // 遍历数组
                        for(var i = 0; i < aihaos.length; i++){
                            var aihaoChk = aihaos[i];
                            aihaoChk.checked = false;//set复选框的选中状态
                        }
                    }
                } */

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

                // 给每一个name="aihao"复选框绑定鼠标单击事件
                for(var i = 0; i < aihaos.length; i++){
                    aihaos[i].onclick = function(){
                        // 在这里控制第一个复选框的选中状态
                        // 第一个复选框选中还是不选中取决于什么？
                        // 所有的aihao复选框的总数量,如果和总选中的数量相同的时候,第一个复选框选中,反之取消选中.
                        var count = aihaos.length; //总数量
                        var checkedCount = 0; //默认选中的数量是0
                        for(var i = 0; i < aihaos.length; i++){
                            if(aihaos[i].checked){
                                checkedCount++;
                            }
                        } // 循环结束之后,所有的被选中的复选框数量就统计完了.
                        // 第一个复选框是选中呢,还是取消选中呢？
                        firstChkElt.checked = (count == checkedCount);
                    }
                }
            }
        </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="firehair"/>烫头
        <br>

    </body>
</html>
```

003-获取一个文本框的value.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>获取一个文本框的value</title>
    </head>
    <body>

        <script type="text/javascript">
            window.onload = function(){
                document.getElementById("btn").onclick = function(){
                    // 获取文本框对象
                    var usernameElt = document.getElementById("username");
                    // 获取value
                    var username = usernameElt.value; // 文本框的value属性用来获取用
户填写的信息.
                    alert(username)
                }
            }
        </script>

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

004-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>
        <!--
            数据库表存储省份和市区的数据
            t_province
            code(pk)     name
            ----------------------
            001          山东省
            002          山西省

            t_city
            code(pk)       name             pcode(fk)
            ------------------------------------------
            1               济南             001
            2               烟台             001

            只要前端浏览器能够获取到山东省的code，假设code=001
            那么后台java程序执行sql语句的时候这样执行：
                select * from t_city where pcode = ?;
```

```
                ps.setString(1, "001");
         -->
         <!-- 这里的this代表当前的下拉列表对象。-->
        <select id="province" 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(){
                document.getElementById("province2").onchange = function(){
                    //这里的this代表的就是当前发生change事件的这个节点对象.
                    //console.log(this.value)
                    console.log(document.getElementById("province2").value)
                }
            }
        </script>

        <select id="province2" >
            <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>
```

005-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">
            /* window.onload = function(){
                document.getElementById("displayTimeBtn").onclick = function(){
                    // 获取系统当前时间,把时间显示到div中
                    var nowTime = new Date();
                    // 显示到div当中
                    document.getElementById("timediv").innerHTML =
nowTime.toLocaleString();
                }
            } */

            /* function display(){
                console.log("显示时间")
            } */
```

```
            // 设置每个1S执行一次display()函数
            //window.setInterval("display()", 1000)

            window.onload = function(){
                document.getElementById("displayTimeBtn").onclick = function(){
                    // 每隔1S调用一次displayTime()函数(设置周期性调用。)
                    // 返回值是一个可以取消周期性调用的value.
                    v = window.setInterval("displayTime()", 1000)
                }

                document.getElementById("stopTimeBtn").onclick = function(){
                    // 停止周期性的调用.
                    window.clearInterval(v)
                }
            }

            function displayTime(){
                var nowTime = new Date();
                document.getElementById("timediv").innerHTML =
nowTime.toLocaleString();
            }

        </script>

        <input type="button" value="显示系统当前时间" id="displayTimeBtn"/>
        <input type="button" value="时间停止" id="stopTimeBtn"/>

        <div id="timediv"></div>

    </body>
</html>
```

006-拼接html的方式，设置table的tbody.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>拼接html的方式，设置table的tbody（非常具有代表性的案例，必须敲5遍。）
</title>
    </head>
    <body>

        <script type="text/javascript">
            /* 从java过来一个json格式的字符串 */
            var fromJava = "{\"total\" : 2, \"students\" : [{\"name\":\"李四
\",\"age\":19},{\"name\":\"王五\",\"age\":18}]}";

            window.onload = function(){
                document.getElementById("displaybtn").onclick = function(){
                    // 解析上面的json格式的字符串,将解析出来的数据放到tbody当中.
                    // 转化json对象
                    window.eval("var json = " + fromJava) //json对象有了.
                    // 设置总记录条数
                    document.getElementById("totalSpan").innerHTML = json.total;
```

```
                    // 拼接HTML
                    var studentArray = json.students;
                    var html = "";
                    for(var i = 0; i < studentArray.length; i++){
                        var s = studentArray[i]
                        html += "<tr>";
                        html += "<td>"+(i+1)+"</td>";
                        html += "<td>"+s.name+"</td>";
                        html += "<td>"+s.age+"</td>";
                        html += "</tr>";
                    }
                    // 将以上拼接的HTML设置到tbody当中
                    document.getElementById("stutbody").innerHTML = html;
                }
            }

        </script>

        <input type="button" value="查看学生信息列表" id="displaybtn" />
        <hr >

        <table border="1px" width="40%">
            <tr>
                <th>序号</th>
                <th>学生姓名</th>
                <th>学生年龄</th>
            </tr>
            <tbody id="stutbody">
                <!-- <tr>
                    <td>1</td>
                    <td>张三</td>
                    <td>20</td>
                </tr>
                <tr>
                    <td>2</td>
                    <td>李四</td>
                    <td>22</td>
                </tr> -->
            </tbody>
        </table>
        总记录条数：<span id="totalSpan">0</span>条
        <!-- 总记录条数：2条 -->
    </body>
</html>
```

007-表单验证.html：

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

        <style type="text/css">
            span {
                font-size: 12px;
```

```
                color: red;
            }
        </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
             （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
             （8）最终表单中所有项均合法方可提交
             */
            window.onload = function(){

                var nameErrorSpan = document.getElementById("nameError");

                // 给id="username"的节点绑定blur事件
                var usernameElt = document.getElementById("username");
                usernameElt.onblur = function(){
                    // 获取用户名
                    var username = usernameElt.value;
                    // 去除掉前后空白
                    username = username.trim();
                    // 用户名不能为空,不能为空串
                    //if(username.length == 0){}
                    if(username == ""){
                        nameErrorSpan.innerHTML = "用户名不能为空";
                    }else{
                        // 用户名不是空,继续判断长度是否合法
                        if(username.length < 6 || username.length > 14){
                            nameErrorSpan.innerHTML = "用户名长度必须在[6-14]之间";
                        }else{
                            // 用户名不为空,并且长度也合法,接下来继续判断用户名中是否有特
殊符号
                            var regExp = /^[a-zA-Z0-9]+$/
                            var ok = regExp.test(username)
                            if(ok){
                                // 合法
                                nameErrorSpan.innerHTML = "";
                            }else{
                                // 不合法
                                nameErrorSpan.innerHTML = "用户名只能由数字和字母组
成";
                            }
                        }
                    }
                }

                // 获得焦点:清空span的错误信息.
                usernameElt.onfocus = function(){
                    nameErrorSpan.innerHTML = "";
                }
```

```
                var pwdErrorSpan = document.getElementById("pwdError");
                // 确认密码失去焦点就验证.
                document.getElementById("confirmpwd").onblur = function(){
                    //获取密码
                    var userpwd = document.getElementById("userpwd").value;
                    //获取确认密码
                    var confirmpwd =
document.getElementById("confirmpwd").value;
                    //进行比对
                    if(userpwd != confirmpwd){
                        pwdErrorSpan.innerHTML = "密码和确认密码不一致";
                    }else{
                        pwdErrorSpan.innerHTML = "";
                    }
                }


                document.getElementById("confirmpwd").onfocus = function(){
                    pwdErrorSpan.innerHTML = "";
                }

                document.getElementById("regbtn").onclick = function(){

                    // 验证用户名,怎么验证用户名？让用户名文本框失去焦点
                    // 重点:使用JS代码怎么触发事件？？？？？？
                    usernameElt.focus(); //触发文本框的获取焦点事件
                    usernameElt.blur();//触发文本框的失去焦点事件

                    // 验证密码,怎么验证密码？让确认密码失去焦点
                    document.getElementById("confirmpwd").focus();
                    document.getElementById("confirmpwd").blur();

                    // 当所有的span都是空的表示表单合法
                    if(nameErrorSpan.innerHTML == "" && pwdErrorSpan.innerHTML
== ""){
                        //提交
                        var formObj = document.getElementById("userForm");
                        // 通过调用submit()方法来完成表单的提交
                        formObj.submit();
                    }
                }
            }

        </script>

        <form id="userForm" action="http://localhost:8080/oa/save">
            用户名<input type="text" name="username" id="username"/><span
id="nameError"></span>
            <br>
            密码<input type="password" name="userpwd" id="userpwd"/>
            <br>
            <!-- 确认密码是不需要提交给服务器的，这个name不要写！ -->
            确认密码<input type="password" id="confirmpwd"/> <span id="pwdError">
</span>
            <br>
            <!-- 表单所有项目都合法才能提交 -->
            <!-- <input type="submit" value="注册" /> -->
            <!-- button不能提交表单，但是JS代码可以提交表单 -->
```

```
            <input type="button" value="注册" id="regbtn"/>
        </form>

    </body>
</html>
```

008-JS中获取元素的三种方式.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS中获取元素的三种方式</title>
    </head>
    <body>

        <div id="div1"></div>
        <div id="div2"></div>

        <input type="checkbox" name="aihao" value="1"/>
        <input type="checkbox" name="aihao" value="2"/>
        <input type="checkbox" name="aihao" value="3"/>
        <input type="checkbox" name="aihao" value="4"/>

        <script type="text/javascript">
            // 这是JS中非常经典的获取元素的三种方式.
            // 根据id获取一个元素
            var div1 = document.getElementById("div1");
            console.log(div1)

            //根据name属性获取多个元素
            var aihaos = document.getElementsByName("aihao");
            console.log(aihaos)

            //根据标签的名字获取
            var divs = document.getElementsByTagName("div");
            console.log(divs)

        </script>

    </body>
</html>
```

1.3 ECMAScript

001-在HTML中嵌入JS代码的第一种方式.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>在HTML中嵌入JS代码的第一种方式：行间事件</title>
    </head>
```

```
    <body>
        <!--
            1、需求：用户点击以下这个按钮，弹出一个对话框，对话框上显示：hello world
            2、JavaScript是一种事件驱动型的编程语言，通常都是在发生某个事件的时候，去执行
            某段代码。其中事件包括很多，例如：鼠标单击事件click，另外还有其它事件，例如：
            mouseover是鼠标经过事件等。并且在JavaScript当中任何一个事件都有对应的事件句柄。
            例如：click对应的事件句柄是onclick，mouseover对应的事件句柄是onmouseover。
            3、所有的事件句柄都是以标签的属性形式存在。例如以下input button就有一个onclick
这样属性。
            只要有用户点击了以下的这个按钮对象，此时按钮对象上发生了鼠标单击事件，那么注册在
            onclick事件句柄当中的JS代码会被执行！onclick后面代码实际上是浏览器负责执行的。
            4、onclick="后面的代码"并不是在浏览器打开的时候执行，浏览器打开的时候，只是将这
个代码
            注册给onclick事件句柄了。等待该按钮的click事件发生，只要发生，后面代码会被事件监
听器
            调用。
            5、怎么使用JS代码弹窗？
                在JS当中有一个内置的BOM对象，可以直接拿来使用，全部小写：window
                其中window对象有一个方法/函数叫做alert，这个函数专门用来弹出对话框！

            6、window.alert('hello world!'); 弹窗的JS代码。
                通过这个代码可以知道：JS中的字符串可以使用单引号括起来，也可以使用双引号。
                JS中的一条语句可以“;”结尾，也可以不以“;”结尾。
        -->
        <input type="button" value="hello1" onclick="window.alert('hello
world!');"/>
        <input type="button" value="hello2" onclick='window.alert("hello
world!")
                                                        window.alert("hello kitty")
                                                        window.alert("你好,中国!")'/>

        <!-- window. 可以省略 -->
        <input type="button" value="hello3" onclick="alert('hello world!');"/>
    </body>
</html>
```

002-在HTML中嵌入JS代码的第二种方式.html：

```
<!-- 脚本块的位置随意，没有限制！ -->
<script type="text/javascript">
    alert("page begin")
</script>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>在HTML中嵌入JS代码的第二种方式：脚本块的方式</title>
    </head>
    <body>

        <!-- 这个按钮会先被加载到浏览器内存。 -->
        <input type="button" value="按钮1" />

        <!-- 脚本块 -->
        <!-- 一个页面中脚本块可以出现多个！ -->
```

```
        <script type="text/javascript">
            /* 在这里直接编写JS代码，这些JS代码在页面打开的时候自上而下的顺序依次逐行执行！
*/
            //alert("hello world"); // 单行注释
            alert("hello zhangsan");
            alert("hello lisi");

            /*
                多行注释
            */
        </script>

        <!-- 最后加载这个按钮2 -->
        <input type="button" value="按钮2" />

    </body>
</html>

<script type="text/javascript">
    alert("page end!")
</script>
```

003-在HTML中嵌入JS代码的第三种方式.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>在HTML中嵌入JS的第三种方式：引入外部独立的JS文件</title>

        <!-- 引入外部独立的CSS文件。这个标签link中属性是href -->
        <link rel="stylesheet" type="text/css" href=""/>
    </head>
    <body>

        <!-- 引入外部独立的js文件 -->
        <!-- script标签引入js文件的时候，是src属性，不是href。 -->
        <script type="text/javascript" src="js/1.js"></script>

        <!-- 引入第二次，这个操作没有意义，测试结果：只要引入一次JS文件，JS文件中的代码就会全
部执行一遍 -->
        <script type="text/javascript" src="js/1.js"></script>

        <script type="text/javascript" src="js/1.js">
            //alert("hello world!~~~~~"); // 这里的代码不会执行！
        </script>

        <!-- 单独的脚本块 -->
        <script type="text/javascript">
            alert("hello world!%%%%%%%%");
        </script>

    </body>
</html>
```

004-关于各种注释.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>关于各种注释</title>

        <style type="text/css">
            /* CSS的注释 */
        </style>

    </head>
    <body>
        <!-- 这是HTML的注释 -->
        <script type="text/javascript">
            /* 这是javascript注释，多行 */
            // 这是javascript注释,单行
        </script>
    </body>
</html>
```

005-标识符和关键字.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>标识符和关键字</title>
    </head>
    <body>
        <script type="text/javascript">

            /* 标识符命名规则和命名规范按照java那一套来就行！ */
            /*
                以下这段代码是JS的for循环，找出其中的关键字和标识符？
                    关键字：var、for
                    标识符：i、alert

                标识符命名规则：
                    标识符只能由数字、字母、下划线、美元符号组成，不能含有其它特殊符号
                    标识符不能以数字开始
                    标识符严格区分大小写
                    关键字不能做标识符
                    标识符理论上没有长度限制

                标识符命名规范？
                    .....
            */
            for(var i = 0; i < 10; i++){
                alert("i = " + i)
            }

        </script>
    </body>
```

```
</html>
```

006-变量.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS的变量</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                回顾Java中的变量？
                    怎么声明？
                        int i;
                        double d;
                        String s;
                    怎么赋值？
                        i = 100;
                        d = 3.14;
                        s = "abc";
                    一行上能声明多个变量吗？
                        int a, b, c = 300;
                        a b c都是int类型。
                        a和b没有赋值。
                        c赋值300

                    重点：Java语言是一种强类型语言，有编译阶段，属于编译型语言。
                        Java语言在编译阶段确定变量的数据类型，也就是说程序还没有
                        运行呢，变量的数据类型就已经确定了，并且该变量的数据类型
                        在这一生中，永远不可变。
                            int x = 1200;【x = true; java的编译器会报错。不让这样做。
语法不对！】

                            double d = x;
                            这行代码表示x的变量中保存的值1200给d变量
                            x变量还是int类型，d变量是double类型。一生不变。

                JS的变量？
                    怎么声明？
                        var 变量名;
                        var i;
                    怎么赋值？
                        变量名 = 值;
                        i = 100;
                    一行上能声明多个变量吗？
                        var a, b, c = 300;
                        声明3个变量，a b c，并且c赋值300，其中a和b变量没有赋值，系统默认赋
值undefined
                        undefined 在JS中一个具体的值，这个值就是 undefined

                    重点：JS语言是一种弱类型语言，没有编译阶段，直接浏览器打开解释执行，
                    在JS中声明变量时不需要指定变量的数据类型，程序在运行过程当中，赋什么
                    类型的值，变量就是什么数据类型，并且变量的数据类型是可变的。
                        var i;
                        i = 100; 到这里i是整数型
```

```
                        i = false; 到这里i就是布尔类型了
                        i = true;
                        i = 3.14;
                        i = new Object();
             */
            var i;
            alert(i) // 变量只声明没有赋值,系统默认赋值undefined, 在JS当中undefined是一个具体存在的值.

            var x = "undefined";
            alert(x) // "undefined"这个不是undefined,它是一个字符串.

            // 声明时,同时赋值
            var k = 100;
            alert(k + 1) // 101

            k = false;
            alert(k);

            k = "abc";
            alert(k)

            // 一行上可以声明多个变量
            //var是一个关键字,就是用来声明变量的,variables单词的前3个字母.
            var a, b, c = 100;
            alert(a) // undefined
            alert(b) // undefined
            alert(c) // 100

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

007-函数初步.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>函数初步</title>
    </head>
    <body>
        <script type="text/javascript">

            //alert(122222);

            /*
                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 int sum(int a,int b){
                            return a + b;
                        }
```

```
                JavaScript当中的函数：
                    函数定义的语法格式是什么？
                        function 函数名(形式参数列表){
                            函数体;
                        }
                        例如：
                            function sum(a, b){
                                return a + b;
                            }

                            函数名：sum
                            形式参数列表是：a和b  (a和b都是变量名！)

                    JS中的函数返回值类型是不需要指定的，因为可以返回任何类型的数据。没有限制。

                    关于JS的的调试？
                        首选方案是：alert()，在程序的某个位置先使用alert弹出某个变量的值，看看是否是你需要的。
                        通过alert可以调试JS代码。

                        另外一种方案是：采用浏览器自带的调试插件，F12
                            F12这个插件中比较重要的面板：
                                控制台
                                查看器
                                网络
                                掌握以上三个面板！！！！！！！！！！！！
             */
            /* function sum(a, b){

            } */

            // 函数必须调用才会执行.
            function sum(x, y){
                alert(x + "," + y);
                //alert("sum函数执行了！");
            }

            //这一切都是因为js是一门弱类型语言!
            sum("abc", false);

            // 调用sum函数
            sum();

            // 调用sum函数
            sum(1);

            // 调用sum函数
            sum(1,2);

            // 调用sum函数
            sum(1,2,3);

            /* JS的函数还有另一种声明方式？*/
            /* function mysum(a, b){

            } */
```

```
            //效果相同
            mysum = function(a, b){
                return a + b; //返回计算结果!
            }

            // 调用函数
            var result = mysum(10, 20);
            alert(result)

            //定义一个函数
            function sayHello(username){
                alert("欢迎" + username + "光临! ")
            }

        </script>

        <!-- 在事件句柄当中调用sayHello函数 -->
        <input type="button" value="hello" onclick="sayHello('李四')" />
    </body>
</html>
```

008-函数初步.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>函数初步</title>
    </head>
    <body>
        <script type="text/javascript">

            // 这里也可以调用sayHello()函数.
            //sayHello();

            // 函数必须手动调用才会执行!
            // 在JS当中函数声明的优先级比较高,打开网页的时候,网页中所有的函数先进行声明.
            function sayHello(){
                alert("hello world!")
            }

            // 这行代码暴露在script标签当中,在页面打开的时候遵循自上而下的顺序依次逐行执行!
            //sayHello();

            //再调用一次
            //sayHello();

        </script>

        <input type="button" value="hello" onclick="sayHello() "/>
    </body>
</html>
```

009-局部变量和全局变量.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>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，叫做全局变量。
                    全局变量在浏览器打开的时候分配空间，直到浏览器关闭的时候才会销毁。
                局部变量：
                    在函数体当中声明的变量，叫做局部变量。
                    局部变量在函数被调用的时候分配空间，函数执行结束之后，内存释放。

                javascript也是遵守就近原则的。

                变量声明了，但是没有手动赋值，系统默认赋值undefined。
                变量没有声明，直接访问，这个时候会报错。（用F12可以查看控制台的错误信息！）

                在javascript当中，如果一个变量声明的时候没有使用var关键字的话，这个变量
                不管是在哪里声明的，都是全局变量。这种全局变量在声明的时候必须手动赋值，不能
                采用系统默认值。
             */
            // 全局变量
            var username = "zhangsan";

            // 定义函数
            function sayHello(){
                // 局部变量
                var username = "lisi";
                alert("welcome, " + username); //就近原则(这是任何一个编程语言都具备
的！)
                // 局部变量
                var i = 100;
            }

            sayHello();

            //在这里访问username
            alert("username = " + username)

            // 在这里访问i变量可以吗？
            // 报错了:i is not defined
            //alert(i);

            var k;
            alert(k) // 声明了一个变量,但是没有手动赋值,系统默认赋值undefined

            // 没有声明这个变量,访问的时候会报错.
            // 报错:kkk is not defined
            //alert(kkk)

            function doSome(){
```

```
                // 声明变量的时候,没有使用var关键字
                email = "zhangsan@123.com";
                //age;
            }

            // 调用函数doSome
            doSome()

            // 访问email变量
            alert("email = " + email)

            //alert("age = " + age)

        </script>

        <input type="button" value="hello btn" onclick="sayHello()"/>

    </body>
</html>
```

010-JS中的函数可以重载吗.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS中的函数可以重载吗</title>
    </head>
    <body>
        <script type="text/javascript">
            // 这个函数已经消失了
            function test1(a, b){
                alert("test1(a, b)")
            }

            // 在JS中函数是不能重载的,也不存在重载机制.
            // JS中的函数只要出现同名函数,之前的函数就消失了.
            // 注意在JS中编写函数名的时候,谨慎一些,以防将其他函数干掉!
            function test1(){
                alert("test1()");
            }

            //test1(1, 2)

        </script>

        <input type="button" value="test" onclick="test1(111, 222)"/>
    </body>
</html>
```

011-JS的数据类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS的数据类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                关于JS的数据类型？
                    1、JS中声明变量不需要指定变量数据类型呀，为什么还要学习数据类型呢？
                        例如：var i = 100; var d = 3.14;
                        在JS中学习数据类型也不是为了声明变量，是为了理解JS程序的底层原理。
                        我们起码要知道100是什么类型，3.14是什么类型，怎么存储的....
                    2、ES6之前JS的数据类型包括6种：
                        Undefined
                        Number
                        Null
                        Boolean
                        String
                        Object

                        其中：Undefined、Number、String、Boolean、Null都属于原始类型（或者叫做基本数据类型）
                        其中：Object属于引用数据类型（或者叫做对象类型）

                    3、在ES6之后引入了其它新的类型，知道就行：
                        Symbol
                        BigInt
                        注意：ES6之后是8种类型。ES6之前是6种类型。
                    4、typeof运算符（非常重要*****）
                        typeof运算符可以在JS代码运行过程当中，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：
                            "undefined"
                            "number"
                            "string"
                            "boolean"
                            "object"
                            "function"
                            并且以上6个字符串都是全部小写的!

                    5、在JS中判断两个字符串是否相等，应该使用“==”,JS中没有equals函数！！！

                    6、关于JS的比较官方的参考文档？
                        https://developer.mozilla.org/zh-CN/docs/Web/API
                        Web API 接口参考

             */
            // sum函数的作用是计算两个数字的和,要求x和y都必须是数字类型,如果不是数字类型应该提示错误信息!
            function sum(x, y){
                // 有一段代码可以动态获取x和y变量的数据类型.
                if(typeof x == "number" && typeof y == "number"){
                    // 求和结果
```

```
                    alert(x + y);
                }else{
                    // 程序执行到这里表示x和y不都是数字
                    alert("对不起，传递的两个参数必须都是数字才可以求和！");
                }
            }

            // i变量声明,但是没有赋值,系统默认赋值undefined
            var i;
            //alert(typeof i) // i变量的数据类型属于Undefined类型,所以typeof运算符的结果是:"undefined"
            console.log(typeof i)

            var k = 3.14;
            //alert(typeof k)

            // 输出信息到控制台,类似于java中的:System.out.println();
            console.log(typeof k)

            var e = 100;
            console.log(typeof e)

            var username = "zhangsan";
            console.log(typeof username)

            var sex = true
            console.log(typeof sex)

            // null属于Null类型,但是typeof运算符的运算结果是"object"
            // 一定要注意:null这个值是基本数据类型,但是typeof运算符的运算结果是"object"
            var v = null;
            console.log(typeof v) // "object"

            var newObject = new Object();
            console.log(typeof newObject) // "object"

            console.log(typeof sum) // "function"

        </script>

        <input type="button" value="计算两个数字的和" onclick="sum('abc', 'def')" />
        <input type="button" value="计算两个数字的和" onclick="sum(1, 111)" />
        <input type="button" value="计算两个数字的和" onclick="sum(1, true)" />

    </body>
</html>
```

012-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：
                    1、只有一个值，它的值就是：undefined
                    2、当一个变量声明之后没有手动赋值，系统默认赋值undefined
                    3、Undefined类型属于原始类型。
             */
            var i;
            console.log(i)

            var k = undefined;
            console.log(k)
            console.log(typeof k)

            var e = "undefined";
            console.log(e);
            console.log(typeof e) // "string"

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

013-Null类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Null类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                Null类型
                    1、Null类型也是属于原始类型。
                    2、Null类型只有1个值：null
                    3、注意：typeof null 运算结果是："object"
             */
            var i = null;
            console.log(typeof i) // "object"
        </script>
    </body>
</html>
```

014-Number类型.html：

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

```
<script type="text/javascript">
    /*
        Number类型
            1、Number类型属于原始类型。
            2、Number类型都有哪些值？
                -1
                0
                1
                2
                3
                3.14
                2.0
                NaN
                Infinity
                ....
            3、NaN？
                Not a Number，表示不是一个数字。
                但NaN一个值。
                它属于Number类型。

            4、什么情况下结果是一个NaN？
                当一个数学表达式的运算结果本应该返回一个数字，
                但是最终无法返回一个数字的时候，结果是NaN。

            5、Infinity是无穷大，当除数是0的时候，最终计算结果是无穷大。

            6、强调：
                JavaScript当中的Number类型代表了java中的：
                    byte short int long float double

            7、在Number类型这一块，有一个函数叫做：isNaN()函数，这个函数
            最终返回布尔类型，返回true表示不是一个数字，返回false表示是一个
            数字。
                isNaN : is Not a Number
                true: 表示不是一个数字
                false:表示是一个数字

                isNaN(数据)：这个函数有一个特点，它会首先尝试将“数据”转换成
                数字，如果转换失败了，则结果就是true。转换为数字成功了，那么
                结果就是false。

                isNaN这个函数是干啥的？
                    就是用来判断“数据”是否是一个数字！！！！！

            8、在Number类型这一块还有一个函数叫做：Number()函数，这个函数的作用可
以将不是
            数字类型的数据转换成数字类型的数据。

            9、parseInt()函数，将字符串数字转换成数字，并且取整。向下取整。

            10、Math.ceil()，这是一个Math工具类中的一个函数，向上取整。

     */
    var k = NaN;
    console.log(typeof k); // "number"

    // 在java中是不允许出现的
    // 在JS中,它可以给你计算出一个惊奇的结果出来,这个结果就是:NaN.
```

```
        // JS检测到 “/” 是一个除号,最终结果肯定是一个数字.
        var result = 100 / "中";
        console.log(result)

        // 在JavaScript当中+可以是字符串连接,也可以是求和.
        var x = 100 + "中";
        console.log(x)

        var v = Infinity;
        console.log(typeof v) // "number"

        // 在java中会报异常.但是在JS中除数可以为0.
        var retValue = 100 / 0;
        console.log(retValue)

        function sum(x, y){
            //console.log(typeof x)
            //console.log(typeof y)
            //console.log(isNaN(x)) // false (true转换为数字结果是1)
            //console.log(isNaN(y)) // false
            if(isNaN(x) || isNaN(y)) {
                alert("对不起, 参与求和的数据必须都是数字! ")
            }else{
                //alert(x + y)
                alert(Number(x) + Number(y))
            }
        }

        //alert(Number("123") + 1)

        console.log(Number("中国")) //NaN


        var s1 = "129.999";
        // 需求:将以上的字符串"123.456",转换成数字,并且只取整数位
        var num = parseInt(s1); // 不会四舍五入,直接取整数位.
        console.log(num + 100)

        var s2 = "123.456呵呵";
        //var s2 = "呵呵123.456"; //这样是无法转换的,最终的结果是:NaN
        var num2 = parseInt(s2);
        console.log(num2) //123

        // 向上取整
        // 使用JS内置的Math这个数学类,Math数学类中有一个函数叫做ceil()
        console.log(Math.ceil(123.001)) //124
        console.log(Math.ceil(123.000)) //124

        // 这个和java不同了,最终结果是浮点数据!
        console.log(10 / 3)  //3.3333333333333335


    </script>

    <input type="button" value="sum" onclick="sum(true, 100)"/>
    <input type="button" value="sum" onclick="sum('中国', 100)"/>
    <input type="button" value="sum" onclick="sum('123', 100)"/>
</body>
```

```
</html>
```

015-Boolean类型.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Boolean类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                Boolean类型：
                    1、Boolean类型属于原始类型。
                    2、Boolean类型只有两个值：true,false，没有其他值。
                    3、Boolean类型中有一个函数：Boolean()函数，这个函数
                    的作用是？
                        将不是布尔类型的转换成布尔类型。

                        转换规律是什么？
                            "只要有东西"结果就是true。
                            有数据就是true，无数据就是false
             */
            console.log(Boolean(1)); // true
            console.log(Boolean(0)); // false

            console.log(Boolean("abc")); // true
            console.log(Boolean("中国")); // true
            console.log(Boolean("")); // false

            console.log(Boolean(NaN)); // false
            console.log(Boolean(Infinity)); // true

            console.log(Boolean(null)); // false
            console.log(Boolean(new Object())); // true

            console.log(Boolean(undefined)); // false

            var i = 0;
            if(i){ // 这里的代码实际上是这样的:if(Boolean(i))
                console.log("不是0");
            }else{
                console.log("是0");
            }

            var i = 10;
            while(i) { // 这里也会自动调用Boolean()函数进行转换.
                alert(i);
                i--;
            }

            // Boolean()函数在JS中会被隐式调用![程序员是不需要手动调用的]
            //var username = "zhangsan";
            var username = "";
            /*
```

```
            if(Boolean(username)) {
                console.log("welcome, " + username)
            }else{
                console.log("对不起用户名不能为空！")
            }
            */
           // 这里不是说if后面小括号可以跟字符串,if后面小括号中永远只能是true或者false
           // 如果不是true,也不是false,那么JS会自动隐式调用Boolean()函数进行类型转换.
           if(username) {
            console.log("welcome, " + username)
           }else{
            console.log("对不起用户名不能为空！")
           }
        </script>
    </body>
</html>
```

016-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、String类型属于原始类型（基本数据类型）
                    2、在JS中怎么定义字符串，包括两种方式：
                        var s = "字符串";
                        var s = '字符串';
                        var s = new String("字符串");
                    3、在JS当中提供了创建字符串的两种方式：
                        如果采用这种方式创建的字符串就属于原始类型！
                            var s = "hello";
                        如果采用这种方式创建的字符串就属于Object类型，这里使用了Object类的子类String，String类是JS内置的，可以直接使用：
                            var s = new String("hello");
                    4、在JS中不管是原始类型的字符串，还是Object类型的字符串，他们的方法和属性都是通用的。
                    5、String当中的常用属性和方法：
                        常用属性：
                            length属性，获取字符串长度
                        常用方法：
                            charAt 方法 获取指定下标位置的字符
                            concat 方法 连接字符串
                            indexOf 方法 获取某个字符串在当前字符串中第一次出现处的索引
                            lastIndexOf 方法 获取某个字符串在当前字符串中最后一次出现处的索引

                            replace 方法 替换
                            split 方法  拆分字符串
                            substr 方法  截取字符串
                            substring 方法  截取字符串
                            toLowerCase 方法 转小写
```

```
                            toUpperCase 方法 转大写
             */
            var s1 = "abc";
            console.log(typeof s1) // "string"

            var s2 = new String("hello");
            console.log(typeof s2) //"object"

            console.log("abcdef".length) //6
            console.log("http://www.baidu.com".charAt(3)) //p
            console.log("abc".concat("def")) //abcdef
            console.log("username=zhangsan&password=123".indexOf("=")) // 8
            console.log("username=zhangsan&password=123".lastIndexOf("=")) //26
            console.log("1980-10-11".replace("-",",")) //1980,10-11 (替换所有需要使
用正则表达式)

            // 拆分字符串返回一个数组.
            var arr = "1980-11-12".split("-"); // var[] arr; 这是错误的,没有这种语法
在JS当中.
            // 遍历数组
            for(var i = 0; i < arr.length; i++){ // arr.length是数组中元素的个数!
                console.log(arr[i]) // 还是使用中括号的方式加下标的方式访问数组中的元素.
            }

            // 对于substr和substring来说,只有1个参数的话没有区别.
            console.log("abcdef".substr(2)) //cdef
            console.log("abcdef".substring(2)) //cdef

            // 对于substr和substring来说,都有两个参数会有什么区别?
            // substr(startIndex, length)
            // substring(startIndex, endIndex),需要注意,不包括endIndex
            console.log("abcdef".substr(2,3)) //cde
            console.log("abcdef".substring(2,3)) //c

            console.log("ABCdef".toLowerCase()) //abcdef
            console.log("ABCdef".toUpperCase()) //ABCDEF

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

017-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、在JS当中内置了一个类型Object，可以将Object类型看做是所有对象的超类/
基类。
                    2、在JS当中默认定义的类型，没有特殊说明的话，默认继承Object。
```

```
                    3、Object类型中有哪些通用属性和方法呢？
                        属性：
                            prototype 属性 | constructor 属性
                        方法：
                            toLocaleString 方法 | toString 方法 | valueOf 方法

                        重点掌握：
                            prototype属性。（prototype翻译为原型）这个属性可以
                            给对象动态扩展属性和方法。

             */
            var obj = new Object();
            console.log(typeof obj)

            // 演示prototype属性
            // 后期给Object类型的对象扩展一个doSome()方法
            Object.prototype.doSome = function(){
                console.log("测试prototype属性！~~~")
            }

            // 后期给Object类型的对象扩展一个username属性.
            Object.prototype.username = "zhangsan"

            // 调用doSome方法
            obj.doSome()

            // 访问对象的username属性
            console.log(obj.username)

            // 可以给String扩展一个方法吗？
            // 给String类型的对象扩展一个mysubstr的方法.
            String.prototype.mysubstr = function(startIndex, length){
                // this表示当前的字符串对象
                return this.substr(startIndex, length);
            }

            console.log("abcdef".mysubstr(2,3)) //cde
            console.log("kingsoft".mysubstr(3,3))  //gso

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

018-JS中怎么定义类new对象调方法.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS中怎么定义类new对象调方法</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、在JS中怎么定义类？包括两种方式
                    第一种方式：
```

```
                        function 类名(形式参数列表){

                            this.属性名 = 参数;
                            this.属性名 = 参数;

                            this.方法名 = function(){

                            }
                        }
                    第二种方式:
                        类名 = function(形式参数列表){

                            this.属性名 = 参数;
                            this.属性名 = 参数;

                            this.方法名 = function(){

                            }
                        }
                 */

                // 既是一个函数,同时又是一个类的定义.函数名是:sayHello,类名是:sayHello
                /* function sayHello(){

                } */

                sayHello = function(){

                }

                // 关键看你怎么调用,如果没有使用new运算符调用,表示普通函数调用.不会在堆中new对象.
                sayHello();

                // 使用new运算符去调用这个函数,显然是把它当做一个类来看待,这个会导致浏览器的堆当中开辟一个新对象!
                var obj = new sayHello(); // obj是一个引用,保存内存地址指向对象!
                var obj2 = new sayHello();

                // 正式的定义一个员工类
                /* function Emp(x, y, z){
                    // 属性
                    this.empno = x;
                    this.ename = y;
                    this.sal = z;
                    // 方法
                    this.work = function(){
                        console.log(this.ename + " is working!!!")
                    }
                } */

                Emp = function(x, y, z){
                    // 属性
                    this.empno = x;
                    this.ename = y;
                    this.sal = z;
                    // 方法
                    this.work = function(){
```

```
                    console.log(this.ename + " is working!!!")
                }
            }

            // 创建对象
            var e = new Emp();
            e.work();

            var e2 = new Emp(111);
            e2.work();

            var e3 = new Emp(2222, "KING");
            e3.work();

            var e4 = new Emp(3333, "SMITH", 800);
            console.log("e4.empno = " + e4.empno) //这个和java相同.
            console.log("e4.ename = " + e4.ename)
            console.log("e4.sal = " + e4.sal)

            /* 访问一个对象的属性还能这样，语法格式：引用["属性名"] */
            console.log("e4.empno = " + e4["empno"])
            console.log("e4.ename = " + e4["ename"])
            console.log("e4.sal = " + e4["sal"])

            e4.work();

            // 给Emp动态的扩展方法呢？
            // 后期动态扩展的方法
            Emp.prototype.getSal = function(){
                return this.sal;
            }

            // 调用扩展的方法.
            console.log(e4.getSal()) //800

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

019-null NaN undefined的区别.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>null NaN undefined的区别</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、==和===有什么区别？
                    == 等同运算符：只比较值是否相等。
                    === 全等运算符：既比较值是否相等，同时又比较数据类型是否相同。
                2、null undefined NaN的区别？
                    类型都是不一样的
                    null和undefined是等同关系。
```

```
             */
            var v1 = true;
            var v2 = 1;

            // == 有点类似于Java语言中的equals方法.
            console.log(v1 == v2) // true

            console.log(v1 === v2) // false

            var v3 = 1;
            console.log(v3 === v2) //true

            // null的类型属于原始类型,typeof运算符的结果是:object
            console.log(typeof null) // "object"
            console.log(typeof NaN) // "number"
            console.log(typeof undefined) // "undefined"

            console.log(null == NaN) //false
            console.log(null == undefined) //true
            console.log(undefined == NaN) //false

            console.log(null === NaN) //false
            console.log(null === undefined) //false
            console.log(undefined === NaN) //false
        </script>
    </body>
</html>
```

020-JS的常用事件.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS的常用事件</title>
        <style type="text/css">
            input {
                border: 1px solid black;
                width: 300px;
                height: 50px;
            }
            #mouseDiv {
                border: 1px solid black;
                background-color: aqua;
                width: 300px;
                height: 300px;
            }
        </style>
    </head>
    <!--
        load事件：
            1、load事件不是在页面加载过程中触发的。
            2、load事件是在页面中所有的元素全部加载完毕之后才发生的。
     -->
    <body onload="console.log('页面加载完毕了!')">
        <script type="text/javascript">
```

```
            /*
                JS的常用事件：
                    （1）blur               失去焦点
                    （5）focus                  获得焦点

                    （3）click                  鼠标单击
                    （4）dblclick           鼠标双击

                    （6）keydown                键盘按下
                    （7）keyup                  键盘弹起

                    （9）mousedown              鼠标按下
                    （10）mouseover             鼠标经过
                    （11）mousemove             鼠标移动
                    （12）mouseout              鼠标离开
                    （13）mouseup           鼠标弹起

                    （16）submit                表单提交
                    （14）reset                 表单重置

                    （15）select                文本被选定
                    （2）change                 下拉列表选中项改变，或文本框内容改变
                    （8）load               页面加载完毕

                提醒：任何一个事件都有对应的事件句柄。事件句柄是在事件名称前添加on就行。
             */
        </script>

        测试选中文本事件：
        <textarea rows="10" cols="30" onselect="console.log('文本被选中了')">
</textarea>
        <br>
        <input type="text" onselect="console.log('文本被选中了')"/>
        <br>
        <br>
        测试change事件：
        <select onchange="console.log('选项被修改!')">
            <option value ="">--请选择您的学历--</option>
            <option value ="gz">高中</option>
            <option value ="zk">专科</option>
        </select>
        <br>
        <br>

        测试表单的提交和重置事件：
        <form action="" onsubmit="alert('表单提交了')" onreset="console.log('表单重
置了')">
            <input type="submit" value="提交"/>
            <input type="reset" value="重置"/>
        </form>

        测试失去焦点事件：<input type="text" onblur="console.log('失去焦点了')"/>
        <br>
        测试获取焦点事件：<input type="text" onfocus="console.log('获取焦点了')"/>
        <br>
        测试click事件：<input type="button" value="click事件"
onclick="console.log('单击')"/>
        <br>
```

```
        测试dblclick事件：<input type="button" value="dblclick事件"
ondblclick="console.log('双击')"/>
        <br>
        测试keydown事件：<input type="text" onkeydown="console.log('keydown....')"
onkeyup="console.log('keyup....')"/>
        <br>
        测试keyup事件：<input type="text" onkeyup="console.log('keyup....')" />

        <br>
        测试mouse相关的时间：
            <div id="mouseDiv" onmouseover="console.log('鼠标经过了')"
                               onmousedown="console.log('鼠标按下了')"
                               onmouseout="console.log('鼠标离开了')"
                               onmouseup="console.log('鼠标弹起了')"
                               onmousemove="console.log('鼠标移动了')"></div>
            <br>




    </body>
</html>

<script type="text/javascript">
    alert("hello world!")
</script>
```

021-JS注册事件的第一种方式.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS注册事件的第一种方式</title>
    </head>
    <body>
        <script type="text/javascript">
            function sayHello(){
                console.log("hello jack!")
            }

            // 这是程序员主动的调用该函数,这个函数不是回调函数.
            //sayHello();

        </script>

        <!-- 这种注册事件的方式就是第一种方式：在标签中使用“事件句柄”，在事件句柄后面编写JS代
码
            当这个事件句柄对应的事件发生之后，“注册”在事件句柄当中的这个代码被监听器调用。

            onclick 鼠标单击的事件句柄，只有当鼠标单击事件click发生之后，注册在onclick后面
的代码
            会被执行。

            以下button标签中的sayHello()函数，在页面打开的时候并不会执行，
            只是在页面打开的过程中完成事件的绑定，完成事件的注册，以后只有
```

```
            当该事件发生之后sayHello()函数才会执行，像这种函数又一种特殊
            的称谓：回调函数。英语单词叫做：callback function

            这个回调函数的特点是：
                监听器负责调用，程序员不负责调用。
                当事件发生之后，监听器会负责调用该函数。
                像这种函数被称为回调函数。callback。

            张三编写的java程序
            public class Test {
                public static void main(String[] args){
                    // 站在“张三”的角度看method()方法属于正向调用。
                    MyClass.method();
                }
            }
            李四编写的java程序
            public class MyClass {
                public static void method(){
                    // 站在method()方法的实现角度来看，我们不负责调用这个method()
                    // 是张三负责调用method()，我们负责实现，此时可以把这个method()叫做回
调方法。
                }
            }
         -->
        <input type="button" value="hello jack" onclick="sayHello()"/>

    </body>
</html>
```

022-小插曲-在HTML中怎么根据id获取节点.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>小插曲-在HTML中怎么根据id获取节点</title>
    </head>
    <body>

        <input type="button" id="mybtn" value="按钮" />

        <script type="text/javascript">
            alert("阻止程序的执行！")
            // 根据id获取元素/节点对象
            // 在JS当中有一个内置的隐含的对象叫做:document
            // document代表整个HTML文档.
            // 在JS当中有一个内置的隐含的对象叫做:window
            // window代表整个浏览器窗口.
            // window对象是BOM的顶级对象,BOM中的老大.
            // document对象是DOM的顶级对象,DOM中的老大.
            // 严格意义上来说,window是包含document的.
            var mybtnElt = document.getElementById("mybtn");

            //console.log(mybtnElt)
            //alert(mybtnElt) //[object HTMLInputElement]
```

```
            // 重点:在JS当中,当你获取了一个节点之后,这个节点中有什么属性你就可以"点"什么.
            //mybtnElt.type = "checkbox";
            //mybtnElt.type = "text";
            mybtnElt.value = "我是一个按钮对象哦! ";

        </script>

    </body>
</html>
```

023-JS注册事件的第二种方式.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS注册事件的第二种方式</title>
    </head>
    <body>

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

        <script type="text/javascript">

            /* 定义一个函数 */
            function sum(){
                console.log("sum function invoke!")
            }

            /* 根据id获取button对象 */
            var hellobtnElt = document.getElementById("hellobtn");

            // 元素中有什么属性,就能"点"什么.
            /* 这行代码在页面打开的时候会执行，这行代码执行的意义是：将sum这个回调函数绑定到
hellobtn的click事件上 */
            /* 这个回调函数sum在什么时候执行？click事件发生之后才会被监听器调用！ */
            //hellobtnElt.onclick = sum  //不要这样写: hellobtnElt.onclick = sum()

            // 回调函数可以是一个匿名函数
            // 这行代码的执行只是完成事件click的注册,给click事件注册一个回调函数.
            // 这行代码执行的时候，回调函数并不会被执行。
            // 只有当这个按钮发生click事件之后,这个回调函数会自动被监听器来调用.
            hellobtnElt.onclick = function(){
                console.log("我是一个回调函数，同时我没有名字，叫做匿名函数! ")
            }

        </script>

    </body>
</html>
```

024-关于代码的执行顺序.html：

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

        <!-- 当前的程序是存在问题的，未解决！！！！！！！！！！！！！！！！！！！  -->
        <script type="text/javascript">

            /* 根据id获取元素 */
            /* 代码执行到这里的时候，id="btn"的节点还没有加载到浏览器的内存当中，所以以下代码返回null */
            var btnElt = document.getElementById("btn");
            //console.log(btnElt) // null

            /* 绑定鼠标单击事件 */
            btnElt.onclick = function(){
                console.log("按钮被点击了，匿名函数被执行！")
            }

        </script>

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

    </body>
</html>
```

025-关于代码的执行顺序.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>关于代码的执行顺序</title>
    </head>
    <!-- <body onload="这里的代码在执行的时候,页面中的元素肯定全部加载完了!"> -->
    <body onload="pageLoad()">

        <script type="text/javascript">
            function pageLoad(){
                // 页面加载完毕之后才会执行这里的代码.此时id="btn"的元素已经加载到浏览器内存了.
                var btnElt = document.getElementById("btn");
                btnElt.onclick = function(){
                    console.log("按钮被点击了，匿名函数被执行！")
                }
            }
        </script>

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

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

026-onload事件句柄.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>onload事件句柄</title>
    </head>
    <!-- 页面加载完毕之后监听器调用myfun()回调函数 -->
    <!-- <body onload="myfun()"> -->
    <!-- <body id="mybody"> -->
    <body>

        <script type="text/javascript">

            /* function myfun(){
                console.log("myfun execute!~~~~~~~~~~~~~~~")
            } */

            // 这种方式可以,就是代码有点多,在JS当中,window对象也可以onload.
            //var mybodyElt = document.getElementById("mybody");
            //mybodyElt.onload = myfun

            /* 在页面打开的时候，以下这行代码会执行，这行代码执行的作用是：将回调函数myfun注册到load事件上 */
            /* 当页面全部加载完毕之后，会发生load事件，load事件发生之后，监听器负责调用回调函数myfun */
            //window.onload = myfun

            // load事件发生之后,后面的“匿名回调函数”会被监听器调用.
            window.onload = function(){
                console.log("hello world!")
            }

            // 页面打开的过程中实际上是各种事件的绑定.
            // 等这些事件一个一个发生之后,回调函数统一都是由监听器来负责调用的.
            /* xxxElt.onclick = function(){}
            xxxElt.onblur = function(){} */

        </script>


    </body>

</html>

<script type="text/javascript">
    alert("页面加载过程中！")
</script>
```

027-事件的最后一个完美的案例.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>事件的最后一个完美的案例</title>
    </head>
    <body>
        <script type="text/javascript">

            /* 这段代码有3个回调函数。 */
            /* 最外层的回调函数是在load事件发生之后才会执行！ */
            window.onload = function(){

                /* //给id="btn1"的元素绑定鼠标单击
                var btn1Elt = document.getElementById("btn1");
                btn1Elt.onclick = function(){ // btn1被单击click之后,这个事件发生了,
才会执行这个回调函数
                    console.log("按钮1被点击了！")
                }

                //给id="btn2"的元素绑定鼠标单击
                var btn2Elt = document.getElementById("btn2");
                btn2Elt.onclick = function(){// btn2被单击click之后,这个事件发生了,才
会执行这个回调函数
                    console.log("按钮2被点击了！")
                } */

                document.getElementById("btn1").onclick = function(){
                    console.log("按钮1单击")
                }

                document.getElementById("btn2").onclick = function(){
                    console.log("按钮2单击")
                }

                document.getElementById("username").onblur = function(){
                    console.log("失去焦点了")
                }

            }

        </script>

        <input type="button" id="btn1" value="按钮1"/>
        <br>
        <input type="button" id="btn2" value="按钮2"/>
        <br>
        <input type="text" id="username" />

    </body>
</html>
```

028-捕捉回车键.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>捕捉回车键（怎么在JS中捕捉键值。）</title>
    </head>
    <body>

        <script type="text/javascript">

            /* function sum(x , y){

            }

            sum();
            sum(1);
            sum(1,2);
            sum(1,2,3);
            */

            /* function myfun(){
                console.log("执行了myfun....没有参数的！")
            } */

            /* function myfun(fdsafdsafds){
                console.log("执行了myfun....有参数的！")
            }

            myfun("abc") */


            // x, y都是变量名,随意的,随便写.只要符合标识符命名规范就行.
            window.onload = function(x){ // x代表的就是load事件对象.
                // 给id="username"的节点绑定keydown事件
                // 后面的这个回调函数是我们负责编写的,但是调用者是监听器.
                // 监听器调用这个回调函数的时候会传过来一个事件对象.
                // 你如果需要使用这个事件对象的时候,你可以写上,你不需要这个事件对象的时候,可以省略.
                document.getElementById("username").onkeydown = function(y){ // y代表的就是一个keydown事件对象.
                    //console.log("keydown.....")
                    // 在这里捕捉键值,当用户敲回车键了,则登录
                    // 新知识点:所有的“键盘事件对象”,有keyCode属性,这个keyCode属性可以获取键值.
                    // keyCode是键盘事件对象的属性.
                    // 记住:键盘上回车键的键值永远都是13.ECS键的键值永远都是27.
                    if(y.keyCode == 13) {
                        console.log("登录，正在进行身份认证，请稍后...");
                    }else if(y.keyCode == 27){
                        console.log("系统安全退出了！")
                    }
                }
            }

        </script>
```

```
        用户名：<input type="text" id="username" />

    </body>
</html>
```

029-void运算符.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>void运算符</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                1、JS当中也有很多运算符
                    算术运算符
                    逻辑运算符
                    关系运算符
                    位运算符
                    三目运算符
                    赋值运算符
                    .....
                    和java那一套是一样的。这里就不再说了。

                2、我们这里主要学习一下JavaScript当中的void和typeof运算符。
                    typeof 运算符：可以在程序运行阶段动态获取变量的数据类型，结果有6个字符
串：
                        "number"
                        "undefined"
                        "boolean"
                        "string"
                        "object"
                        "function"

                    void运算符：
                        语法格式：
                            void(表达式)
                            执行表达式，但不返回任何结果。
                            即使表达式有执行结果，最终经过void运算之后就什么都没了。
             */
        </script>

        页面顶部！
        <br>
        <input type="button" value="按钮" onclick="alert('hello world')" />
        <!-- 添加javascript:表示告知后面是一段JS代码，大部分情况下javascript: 是可以省略
的！ -->
        <input type="button" value="按钮" onclick="javascript:alert('hello
world')" />

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

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

        <!--
            1、添加href属性，这样可以保留住超链接的样式。
            2、onclick后面可以编写JS代码，这样点击的时候JS代码执行了。
            3、执行完JS代码还必须让页面不跳转!

            关键的位置是：怎么保证不跳转呢？
                如果没有跳转路径的话,它自然就不跳转了。

            href="javascript:void(0)"
                void运算符执行表达式，但是不返回任何结果：void(表达式) 执行结束之后啥也没
有，并且也没有 ""

                这样记忆：
                    href后面添加一个：javascript:void(0)
                    表示把href的连接地址废弃掉。
         -->
        <a href="javascript:void(0)" onclick="alert('执行JS代码了!')">要求：保留住超
链接的样式，并且点击我执行一段JS代码，执行完JS代码之后保证页面不跳转! </a>

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

030-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、JS中的控制语句，和java相同，JS中除了java的控制语句之外，还有一些特别的：
                    选择结构：
                        if语句
                        switch语句
                    循环结构：
                        for
                        while
                        do..while
                    转向语句：
                        break
                        continue
                        return
                2、JS有哪些特殊的语句呢？
                    了解一下，不需要掌握：
                        for..in 语句
```

```
                with    语句
 */
for(var i = 0; i < 10; i++){
    console.log("i = " + i)
}

var username = "";
if(username) { // Boolean()函数
    console.log("welcome," + username)
}else{
    console.log("username不能为空! ")
}

// 数组(JS中的数组对象创建的时候使用的是：中括号)
//var arr = [1,24,5,65,76,7];
// 在JS的数组中数据的类型随意,可以不一致.
var arr = [true,"abc",5,false,76,3.14];

// 对数组遍历
/* for(var i = 0; i < arr.length; i++){ //数组有length属性.
    console.log(arr[i])
} */

// for..in语句
for(var fdsafdsafds in arr){ //arr是数组的话,fdsafdsafds就是数组的下标.
    //fdsafdsafds是数组的下标
    //console.log(fdsafdsafds)
    console.log(arr[fdsafdsafds])
}

// for..in语句还可以取对象的属性值.
Employee = function(empno,ename){
    this.empno = empno;
    this.ename = ename;
}

var e = new Employee(7369, "SMITH");
console.log(e.empno + "," + e.ename)
console.log(e["empno"] + "," + e["ename"])

// for..in语句遍历对象的属性
for(var fdsafdsa in e){ // e是JS对象的话,fdsafdsa就是对象的属性名.并且属性名是字符串.
    //console.log(fdsafdsa)
    //console.log(typeof fdsafdsa) //string
    console.log(e[fdsafdsa])

    // 这种方式就不行了.
    //console.log(e.fdsafdsa)
}

var x = new Employee(7369, "SMITH");
//console.log(x.empno + "," + x.ename)
// with语句.
with(x) {
    console.log(empno + "," + ename)
    console.log(empno + "," + ename)
    console.log(empno + "," + ename)
```

```
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
                console.log(empno + "," + ename)
            }

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

031-JS的内置对象-Array.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Array是一个数组类型</title>
    </head>
    <body>
        <script type="text/javascript">
            /*
                关于JS中的数组
                    1、怎么创建数组对象
                    2、怎么遍历数组
                    3、怎么修改数组中的某个元素
                    4、怎么读取数组中的某个元素
                    ...

                java语言创建数组？
                    String[] strs = {"abc", "def", "xyz"};

                JS语言创建数组？
                    var strs = ["abc", "def", "xyz"];
             */

            // 创建数组的第一种方式
            var a = []; //创建一个长度为0的数组对象

            // JS中数组中的元素类型可以不一致.
            var a2 = [12,3,4,5,6,false,true];

            // JS中数组的长度是可变的.
            console.log(a2[0])
            console.log(a2[a2.length - 1]) // 数组有length属性获取数组中元素的个数.

            // 没有扩容之前:[12,3,4,5,6,false,true]
```

```
            // a2[100]会导致自动扩容:
[12,3,4,5,6,false,true,undefined,undefined,undefined....., 1234]
            // JS中数组没有下标越界这一说!
            a2[100] = 1234;
            console.log("数组的长度=" + a2.length) //数组的长度=101 (下标0到100，正好
101个元素)
            console.log(a2[100]) // 1234
            console.log(a2[99]) // undefined
            console.log(a2[-100]) // undefined
            console.log(a2.length) //101

            // 创建数组的第二种方式
            var arr1 = new Array(); // 创建长度为0的数组
            console.log(arr1.length) //0

            var arr2 = new Array(3); // 创建长度为3的数组
            console.log(arr2.length) // 3

            for(var i = 0; i < arr2.length; i++){
                // 3个undefined
                console.log("---->" + arr2[i])
            }

            var arr3 = new Array(1, 2, 3, 54, 67, 7); // 创建有指定值的数组
            console.log(arr3.length) // 6

            for(var i = 0; i < arr3.length; i++){
                console.log("$$$$$$=====>" + arr3[i])
            }

            arr3[3] = 45;

            for(var i = 0; i < arr3.length; i++){
                console.log("******=====>" + arr3[i])
            }


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

032-Array的常用方法.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Array的常用方法</title>
    </head>
    <body>
        <script type="text/javascript">
            // 创建一个长度为0的数组对象
            var a = [];

            console.log("数组的元素个数是：" + a.length)
```

```
            // 添加元素
            a.push(100);
            a.push(200);
            a.push(300);
            a.push(10);
            a.push(20);

            console.log("数组的元素个数是：" + a.length)

            a = [1,2,3];
            // push方法:向数组中添加一个元素,并且加到末尾.
            a.push(false);
            for(var i in a){
                console.log(a[i])
            }
            console.log(a.length) //4
            // pop方法:将数组末尾的元素弹出,并且数组长度-1
            console.log(a.pop()) //false
            console.log(a.length) //3

            // 注意:数组的push和pop方法联合起来,实际上是模拟了栈数据结构!
            var arr = [];
            arr.push(1)
            arr.push(2)
            arr.push(3)

            console.log(arr.pop())
            console.log(arr.pop())
            console.log(arr.pop())

            // 翻转数组
            var array = [1,5,65,6,67];
            // 翻转
            array.reverse();
            // 遍历
            for(var i in array){
                console.log("%%%%====>" + array[i])
            }

            // 连接(将数组中的每一个元素以“$”连接成一个字符串)
            //var str = array.join("$")
            var str = array.join("-")
            console.log(str) //"67-6-65-5-1"

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

033-JS内置对象Date.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JS内置对象Date</title>
    </head>
```

```
    <body>
        <script type="text/javascript">
            /*
                JS的内置对象：Date
            */
            var time = new Date(); // 获取系统当前时间的.
            //console.log(time.getYear()) //120 (getYear()方法过时了，不建议用了，获取的年是两位的。1985最终获取之后是85)

            console.log(time) //Date Thu Mar 26 2020 17:55:13 GMT+0800 (中国标准时间)

            // 进行格式转换
            // 可以将年月日的信息都拿出来.然后自己拼接格式.
            var year = time.getFullYear();
            var month = time.getMonth(); // 0-11 表示 1-12
            //var day = time.getDay(); // 获取的是星期几
            var day = time.getDate(); // 获取一个月份中的第几天
            console.log(year + "年" + (month + 1) + "月" + day + "日")

            // 获取时分秒毫秒
            /* time.getHours()
            time.getMinutes()
            time.getSeconds()
            time.getMilliseconds() */

            //在JS中提供了一个函数toLocaleString(),其实这个函数是Object中的.
            // 转换成具有本地语言环境的日期格式
            var strTime = time.toLocaleString();
            console.log(strTime) //2020/3/26 下午5:59:55

            // 怎么获取自"1970年1月1日 00:00:00 000"到系统当前时间的总毫秒数.
            var now = new Date();
            var timeMillis = now.getTime(); //这个getTime()方法是一个重点方法.
            console.log(timeMillis)

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

034-BOM和DOM的关系.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>BOM和DOM的关系</title>
    </head>
    <body>

        <div id="div1">

        </div>

        <script type="text/javascript">
            /* BOM是浏览器，DOM是浏览器上的网页，所以BOM是包含DOM的。 */
```

```
            window.onload = function(){

                console.log(window.document)
                console.log(document)

                // window对象是BOM的顶级老大
                // document对象是DOM的顶级老大
                // 实际上完整的写法是: window.document, 只不过window. 可以省略.
                //var divObj = window.document.getElementById("div1")
                var divObj = document.getElementById("div1")
                console.log("========>" + divObj)
            }
        </script>
    </body>
</html>
```

035-eval函数.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>eval函数</title>
    </head>
    <body>
        <script type="text/javascript">

            // eval函数可以将一个字符串当做一段JS代码解释执行!
            window.eval("var i = 100")

            //var i = 10011
            console.log(i)

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

036-怎么创建JSON对象以及访问JSON对象的属性.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body>
        <script type="text/javascript">
            /*
            1、什么是JSON?
                JavaScript Object Notation（JavaScript标记对象）
                简称JSON。
                JSON是一种轻量级的数据交换格式。

                什么是轻量级:
```

体现在JSON的体积小。虽然一个小的体积可能表示的数据很多。

什么是数据交换：
C语言和Java语言之间交换数据，
python和Java语言之间交换数据，
javascript和java之间交换数据。

透露一下：在现代的开发中，能够做数据交换的，包括两个：
第一个：JSON
第二个：XML
JSON和XML都是非常标准的数据交换格式。
XML体积大，解析难度大。
JSON体积小，解析更容易。
XML和JSON相对比来说，XML的语法严格，json的语法相对松散。

2、在JavaScript当中，json是以对象的形式存在的。

3、在javascript当中，怎么定义JSON格式的对象，怎么访问对象的属性呢？
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,
        "属性名" : 属性值,
        "属性名" : 属性值,
        ....
}
```

注意：属性值，可以是任意类型。
JSON是一种无类型的对象，直接一个大括号包起来就是一个JSON对象了。

4、注意：在JS中[]和{}有什么区别？

```
    [] 是数组对象
    {} 是json对象
 */
var emp = {
    "empno" : 7369,
    "ename" : "smith",
    "sal" : 800
}

// 怎么访问对象的属性？
// 第一种方式
console.log(emp.empno)
console.log(emp.ename)
console.log(emp.sal)

// 第二种方式
console.log(emp["empno"])
console.log(emp["ename"])
console.log(emp["sal"])

var person = {
    "name" : "zhangsan",
    "sex" : false,
    "aihao" : ["抽烟","喝酒","烫头"]
}
```

```
            console.log(person.name)

            console.log(person.sex ? "男" : "女"),

            var aihaos = person.aihao;
            for(var i = 0; i < aihaos.length; i++){
                console.log(aihaos[i])
            }

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

037-JSON对象的属性值可以是JSON对象吗.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JSON对象的属性值可以是JSON对象吗</title>
    </head>
    <body>
        <script type="text/javascript">
            // json对象1
            /* var addr = {
                "city" : "北京",
                "street" : "大兴"
            } */

            // json对象2
            /* var user = {
                "username" : "zhangsan",
                "password" : "123",
                "email" : "zhangsan@123.com",
                "address" : addr
            } */

            var user = {
                "username" : "zhangsan",
                "password" : "123",
                "email" : "zhangsan@123.com",
                "address" : {"city" : "深圳","street" : "宝安"}
            }

            // zhangsan住在哪个城市怎么访问？
            console.log(user.username + "居住在" + user.address.city)
        </script>
    </body>
</html>
```

038-设计一个JSON格式的数据可以表示全班人数和每个学生信息.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>设计一个JSON格式的数据可以表示全班人数和每个学生信息</title>
    </head>
    <body>
        <script type="text/javascript">
            //设计一个JSON格式的数据可以表示全班人数和每个学生信息
            var students = {
                "total" : 3,
                "data" : [{"name":"陈赓", "age":20},{"name":"吴雨阳","age":21},
{"name":"殷远庭", "age":23}]
            };

            // 理解了吗？JSON很容易解析,一顿"点"就行.
            console.log(students.data[0].name)

            // 访问以上的json对象,将总人数取出,将每个学生的信息取出
            console.log("总人数：" + students.total)

            // 访问每一个学生数据
            var arr = students.data;
            for(var i = 0; i < arr.length; i++){
                var s = arr[i];
                console.log("姓名：" + s.name + ",年龄：" + s.age)
            }

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

039-java传过来的是一个字符串怎么变成json对象呢.html：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>java传过来的是一个字符串怎么变成json对象呢</title>
    </head>
    <body>

        <script type="text/javascript">
            /*
                1、java和javascript两个语言怎么交换数据呢？
                    可以使用JSON格式的字符串。
                    JSON就是一种轻量级的数据交换格式。

                2、java的JDBC连接数据库查询数据，然后将数据拼接成JSON格式的字符串，
                将JSON格式的字符串传给javascript，然后在javascript当中把json格式
                的字符串转换成JSON对象，这样就可以从json对象中取数据了，这样就完成了
```

```
                数据的交换。
             */

            // json对象
            /* var json = {
                "name" : "zhangsan",
                "age" : 20
            } */

            // 双引号当中的是一个普通的不能再普通的字符串,这个字符串是java给我们浏览器的.
            var fromJavaJSON = "{\"name\":\"zhangsan\", \"age\":20}";  //这个不是json对象,是一个字符串.

            // 你需要将json格式的字符串转换成json对象.
            // eval函数的作用是:将后面的字符串当做一段JS代码解释并执行.
            window.eval("var stu = " + fromJavaJSON) //重点中的重点,这个可以将json格式的字符串转换成json对象.

            // 上面代码执行结束之后,等同于这里创建了一个json对象.
            /* var stu = {
                "name" : "zhangsan",
                "age" : 20
            }; */

            // 转换成json对象的目的是为了取数据.(这样javascript和java之间两个不同的编程语言就完成了数据的交换!)
            console.log(stu.name + "," + stu.age)

            window.eval("var i = 0")
            alert(i)
        </script>

    </body>
</html>
```

040-正则表达式.html:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>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？
                    正则表达式是一门独立的学科，基本每个语言都支持。
                    正则表达式不是JS专属的。不过在JS中使用居多。
                    通常使用正则表达式进行字符串格式匹配。

                    正则表达式是有一堆特殊的符号组成的一个表达式。
                    每一个特殊的符号都有特殊的代表含义。

                    例如：
                        qq号的正则表达式。
```

邮箱地址的正则表达式。

邮箱地址格式验证：

程序中有一个邮箱地址的正则表达式。

用户输入了一个邮箱地址。

那么邮箱地址的正则表达式和邮箱地址进行匹配，能匹配成功，表示合法，反之表示不合法。

2、对于javascript程序员来说，我们对于正则表达式掌握到什么程度呢？

第一：能够看懂正则表达式

第二：简单的正则要会写

第三：要能够独立的从网络当中搜索到你想要的正则表达式（搜索能力要有）

第四：要会创建JS的正则表达式对象。

第五：要会调用JS正则表达式对象的方法。

3、常见的正则表达式符号有哪些？

. 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 0-N次

+ 重复一次或更多次 1-N次

? 重复零次或一次 0或1次

{n} 重复n次 n次

{n,} 重复n次或更多次 n+次

{n,m} 重复n到m次 n到m次

注意：数量永远匹配的都是前面的那个字符出现的次数。

\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

| 表示或者

[a-z]{1} a到z所有的字符中的任意1个。

[a-zA-Z0-9]{3,} 前面这堆中的任意字符至少出现3个。

[1-9][0-9]{4,} qq号的正则表达式，最小的qq号是10000

[1-9] 没有指定数量的时候，默认是1个。

4、邮箱的正则表达式：

^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$

这个邮箱地址从网上找了之后不一定能用，你需要测试。反复测试。

5、在JS中怎么创建正则表达式对象呢？

包括两种方式，重点使用第一种

第一种方式：直接量语法

```
                        var regExp = /正则表达式/标记

                    第二种方式：使用内置类RegExp类。
                        var regExp = new RegExp("正则表达式", "标记")

                    标记是可选项！！！！！都有哪些值可选呢？
                        g：全局 global
                        i: 忽略大小写 ignorecase
                        gi: 全局扫描，并且忽略大小写。

                6、正则表达式对象有一个很重要的方法：
                    var emailRegExp = /^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+
([-.]\w+)*$/
                    var ok = emailRegExp.test("用户输入的字符串");
                    返回值ok是true表示，匹配成功了。

                    其实在JS中，字符串String也是支持正则表达式的。
             */
            // 字符串对象使用正则表达式
            console.log("1980-11-10".replace("-", "/"))
            // g表示global全局的,所有的 - 替换成/
            console.log("1980-11-10".replace(/-/g, "/"))

            function checkEmail(){
                //获取邮箱地址
                var email = document.getElementById("email").value;
                // 创建正则表达式对象
                var regExp = /^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$/;
                // 验证
                var ok = regExp.test(email);
                if(ok){
                    alert("邮箱地址合法")
                }else{
                    alert("邮箱地址不合法");
                }
            }

        </script>

        邮箱地址：<input type="text" id="email" />
        <input type="button" value="验证邮箱地址" onclick="checkEmail()"/>
    </body>
</html>
```