2014 年辛星 web 前端教程贺岁版

第四本：Javascript 教程

**********2014 年辛星 web 前端教程贺岁版**********

1.我最早萌生出要写教程的时间是在 2014 年夏季，当时写了一个 120 页左右的教程，看到有不少读者喜欢，于是我进行了扩充，就有了贺岁版。

2.贺岁版在内容上更加充实，格式上更加优美，书单如下：

编号	书名	是否完工
1	html 基础	已完工
2	css 基础	已完工
3	css 典型应用	已完工
4	Javascript 基础	已完工
5	html5 基础	已完工
6	css3 基础	已完工
7	jQuery 基础	已完工
8	jQuery UI 基础	已完工
9	Bootstrap 基础	已完工

3.自己写的书就像自己的孩子一样，写这些资料不图什么，就是希望能够给朋友们带来一些帮助。

4.我们相信它能做的更好，因此我们特别期待您能够参与进来，给出自己的建议，帮助我们把它做的更好。

****

**********我们的平台**********

我们的 QQ 群：392328871

我们的贴吧：辛星

让我们共同构建一个学术的净土，梦想的天堂。

****

**********致辞**********

1.百度搜索“辛星　web 前端”相信您会看到更多。

2.辛星，期待您的关注。

****

**********寄语**********

特色：更新更全更实用。　　目标：专业权威可信赖。

纲领：传播编程知识，振兴中户软件。

前进的道路，辛星陪伴您。　　只要星哥在，编程充满爱。

****

**********加入我们**********

我们的 QQ 群：392328871

我们的贴吧：辛星

我的个人 QQ：1808347923

我的个人邮箱：xinguimeng@163.com

我的个人博客：blog.csdn.net/xinguimeng

****

**********与君共勉**********

1.每周五都会有新一批的教程出现，风雨中，我们一直在努力。

2.当学习成为一种习惯，进步就成为一种必然能。

3.我们是一个团结友爱奋进的大家庭，愿意加入我们吗？

****

**********纲领**********

传播编程知识，振兴中华软件。我心永恒，始终如一。

****

****

目录：

****

第零部分：提纲挈领

第一节：简介 Javascript

>>>>>>>>>>小吹一下<<<<<<<<<<<<<<<<<<<<

1.要说到 Javascript 的火爆程度，恐怕还真的没有几门语言敢与之比肩，至少目前是这样。

2.它已经不仅仅是那个仅仅为了验证下表单、做个弹窗的玩具语言了，而是一个能够跻身于 web 前端、后端、甚至连 PDF 阅读器上都支持 Javascript，可以说，js 的使用会越来越广泛。

3.当然啦，咱们不黑不吹，不过 Javascript 的前景绝对不仅仅是用来在前端写个网页特效那么简单。

4.不过单纯就网页的动态性上，Javascript 就已经成为事实上的标准了，其霸主地位谁能撼动？目前至少无人可以撼动其根基。

>>>>>>>>>历史 <<<<<<<<<<<<<<<<<<<<<<<<

1.首先 Javascript 是一门轻量级的解释型脚本语言，和其他语言一样，它也是弱类型的。

2.它最开始是在 1995 年由 Netscape 公司的 Brendan Eich 首次设计实现而成，最初名字叫做 LiveScript。

3.在 Netscape 公司于 Java 语言的创始公司 Sun 公司合作时，为了能够“让它看上去更像 Java”，它被更名为 Javascript，这也是为什么它取了这么一个令人混乱的名字的原因。

4.在 1997 年，在 ECMA(欧洲计算机制造商协会)的协调下，由 Netscape、Sun、MicroSoft、Borland 等公司制定了一个统一的标准：ECMA-262，因此，Javascript 也可以成为 ECMAScript。

5.咱们这里说的 Javascript，并不是真正意义上的 Javascript 语法，而是混杂了 BOM、DOM 与 ECMAScript 语法的一个大杂烩。

>>>>>>>>>>个人感触 <<<<<<<<<<<<<<<<<<<

1.作为一门脚本语言，本来就应该是简单易学易用的。

2.但是，我感觉它并不算是一门简洁的语言，而且我还感觉它是一门很混乱的语言。

3.它或许不算难，但是我依然感觉它很乱，而且感觉不够规范。

4.这里咱们不接触太底层的东西，主要就是学习下它的语法和常见应用以及一些高级特性。

>>>>>>>>>>>>开发前的准备<<<<<<<<<<<<<<<<<<

1.和前面一样，我们只需要准备的就是一个文本编辑器和一个浏览器。

2.对于文本编辑器来说，我在写教程的时候喜欢用那些简单小巧的，这里选择也是以 sublime 为主。

3.对于浏览器，我个人比较喜欢 Chrome、Firefox 和 Opera，当然首选自然是火狐啦。

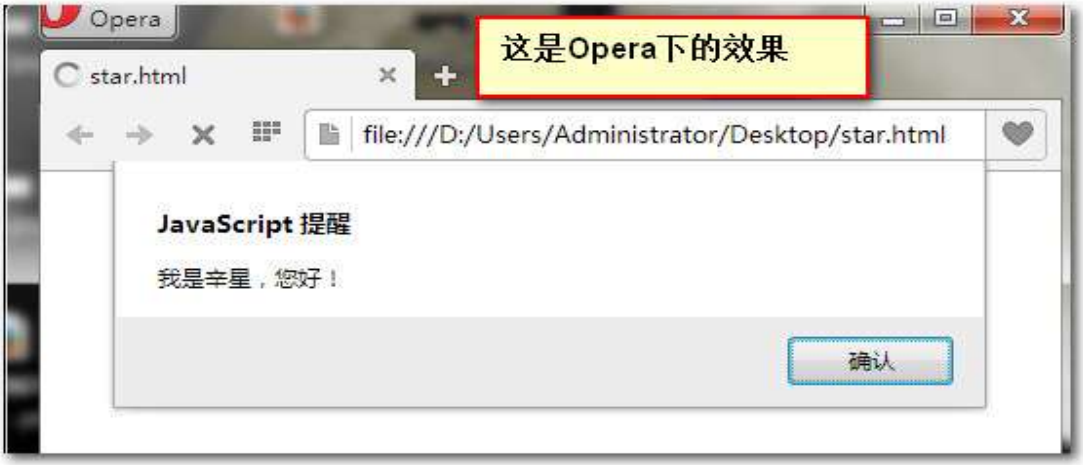
>>>>>>>>>>>>>第一个程序<<<<<<<<<<<<<<<<<<<

1.首先我们在浏览器中写入如下代码：

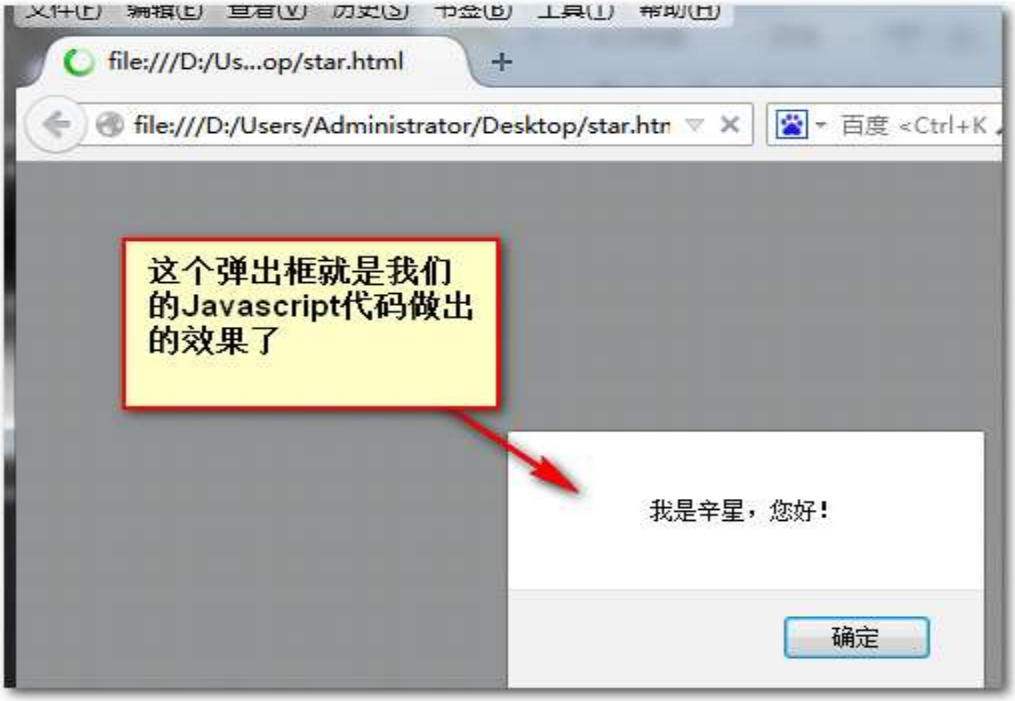
star.html

```
<html>
    <body>
        <script type="text/javascript">
            alert("我是辛星，您好！");
        </script>
    </body>
</html>
```

2.然后我们看下在 Opera 浏览器下的效果吧：



3.当然不同的浏览器的效果可能略有不同，不过性质都差不多，下面是火狐浏览器的效果：



4.这里我们分析一下源代码吧，这里的<script>标签表示其内部写的是 Javascript 代码，通常来说我们可以放到 body 元素的尾部，这个习惯的由来是之前网页加载较慢的时候，一般都是先让网页加载完毕，然后再去执行 Javascript 代码。

5.这里的 alert("我是辛星，您好！")；这行代码的含义就是抛出一个弹窗，弹窗的文字就是“我是辛星，您好！”。

>>>>>>>>>>>>>js 文件和 html 文件的分离<<<<<<<

1.通过前面的学习，我们知道 html 文件和 css 文件是可以分开的。

2.当然啦，js 文件和 html 文件也是可以分开的，我们使用 script 这个元素来导入 js 文件就可以了，我们给该元素的 src 属性赋值就是该 js 文件的地址。

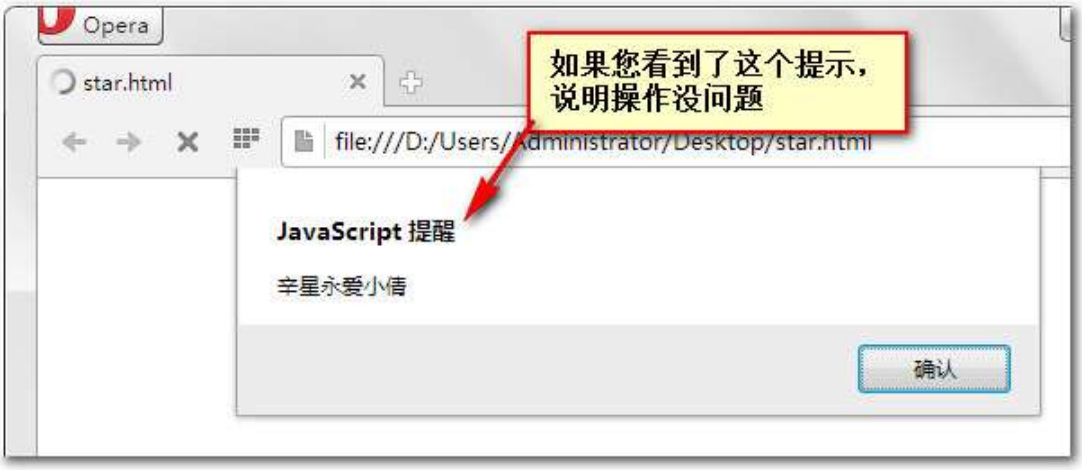
3.那么这个 script 元素是不是依旧写到 body 元素中呢？答案是可以的。当然，我们也可以把它放到 head 元素中去，也就是我们一般在导入完毕 css 文件之后就导入 js 文件。

4.我们就先在该目录下新建一个 star.html 文件和 star.js 文件，我们来个截图看看效果吧：

5.我们在 star.js 中写入如下内容：

6.然后我们在 star.html 中写入如下内容：

7.然后我们看看效果吧：



8.当然有人可能会问：此时把 script 元素放到 head 元素中可以吗？显然是可以的，比如代码如下：

```
<html>
    <head>
        <script src="star.js" ></script>
    </head>
</html>
```

9.其实都可以放到 html 元素外面去，都是可以的，比如代码如下：

```
<script src="star.js" ></script>
<html>
    <head>

    </head>
</html>
```

10.不过讲这些就有点没意思了，虽然它们在执行上是没错的，但是不是很推荐，毕竟代码是给大家看的嘛。

>>>>>>>控制台<<<<<<<<<<<<<<<<<<<<<<<<

1.可能有写些基础的朋友们告诉我说：是不是在控制台写 Javascript 代码更纯粹些？

2.答案确实是这样的，我给朋友们个范例吧：

3.当然啦，不过我们作为一个入门教程，这里并不会讲的太深，因此我们并不会在这里写代码。

4.我们还是采用最传统的那种在文本文件中写代码，然后让浏览器去执行的那种方式，虽然很原始，但是很有效，不是吗？

>>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<<<<<<

1.本小节我们并没有介绍太多复杂的东西，因此朋友们看起来应该还是蛮轻松的。

2.我们介绍了两种写 Javascript 代码的方法，我们后面如无特殊说明，都会采用第二种方式，也就是单独写到一个 js 文件中去。

3.当然啦，Javascript 是一门博大精深的语言，仅仅靠一本书想讲的非常透彻，是不现实的，但是本书可以让您领略到 Javascript 比较基础的，比较常用的那部分内容。

第二节：必看

>>>>>>>>>前言<<<<<<<<<<<<<<<<<<<<<<<<

1.说实话，我很少写这种“必看”之类的，我们整个 web 前端教程，一般都是全书都是第 xx 节这种格式。

2.但是 Javascript 太特殊了，特殊到我必须专门拿出来一节介绍一下它的学习方式。

>>>>>>>>>js 的学习方式<<<<<<<<<<<<<<<<<<<<

1.为了达到真正意义上的“按需来学”的目标，我必须介绍一下我们的需求。

2.如果朋友们特别想学好 Javascript，那么这本书是不够的，那么建议去阅读我们的 Javascript 系列教程(现在还没写，会考虑写的)，它会包含一系列的 Javascript 教程，希望会对您的学习有所帮助。而本书的第一部分也会对您的入门有一个不错的提高。

3.如果朋友们只是想做个前端的特效，那么可以抛弃本书的第一部分，直接从 BOM 或者 DOM 开始看起就可以了。

4.如果朋友们学习 Javascript 是为了做其他东西，或者做和 web 无关的东西，那么建议放弃本书，直接阅读我们的 Javascript 系列教程，希望对您有所帮助。

5.我们不希望浪费您的时间，因此我们认为本节是必须要读的。

>>>>>>>>>>回馈我们<<<<<<<<<<<<<<<<<<<<<

1.如果您感觉我讲的不够清楚，那么也可以发送邮件给我，来及时联系我，可以重新分类。

2.如果您对本书有任何的意见，都可以及时联系我奥。

>>>>>>>>>>>>>>>致辞<<<<<<<<<<<<<<<<<<<<

1.我用我的努力给您的成功提供助力。

2.传播编程知识，振兴中华软件，我心永恒，始终如一。

3.饿死不收费，累死不停更，我们从未停息。

4.每周更新，期待您的关注，我是辛星。

第一部分：Javascript 语法简介

第一节：变量、数据类型、运算

>>>>>>>>>>>>>变量<<<<<<<<<<<<<<<<<<

1.所谓变量，就是一个用于存储信息的“容器”。

2.每个变量有一个唯一的名字，用于彼此区分。

3.下面说一下变量名的三个基本原则：①变量名必须以字母开头，后面跟数字、字母和下划线。②虽然也可以以$和_开头，但是不推荐。③变量名称区分大小写。

4.比如 a、xin、qian 都是合法的变量名，且 a 和 A 是两个不同的变量名。

5.每个变量都会有一个变量类型。

>>>>>>>>>>>>数据类型<<<<<<<<<<<<<<<<<<

1.Javascript 的数据类型主要分为如下几种：

类型名	汉语名
String	字符串类型
Number	数值类型
Boolean	布尔类型
Array	数组类型
Object	对象类型
Null	空类型
Undefined	未定义类型

2.我们在随后的学习中会逐步的学习它，因此朋友们不必害怕。

>>>>>>>>>>字符串类型<<<<<<<<<<<<<<<<<<

1.所谓字符串类型就是一串字符，比如“辛星”，比如“Star”。

2.字符串可以使用单引号，也可以使用双引号引起来。

>>>>>>>>>>数值类型<<<<<<<<<<<<<<<<<<

1.数值类型就是生活中常见的各种数据了，比如 5，3.2 等等。

2.我们还可以用科学计数法来表示，看下面示例：

准则：　　xey 表示 x 乘以 10 的 y 次方

示例：　　2e5 表示 200000

>>>>>>>>>>布尔类型<<<<<<<<<<<<<<<<<<<<

1.布尔类型也就是逻辑类型，它通常用于逻辑运算。

2.它只有两个值：true 和 false，分别表示真和假。

>>>>>>>>>>变量操作<<<<<<<<<<<<<<<<<<<<<<

1.我们可以声明一个变量，使用 var 变量名； 这种格式，如果是声明多个变量，使用 var 变量名 1，变量名 2，....变量名 n；这种格式，多个变量名之间用逗号分开。

2.我们还可以给变量赋值，使用变量名＝值；这种格式。

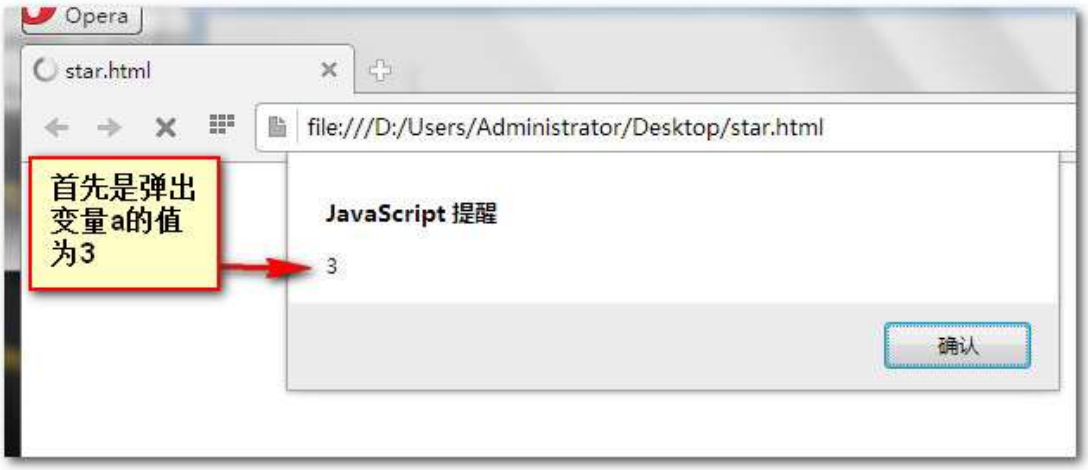
3.我们可以给一个变量进行声明的同时赋值，也就是使用

　　var 变量名 ＝ 值；　　这种格式。

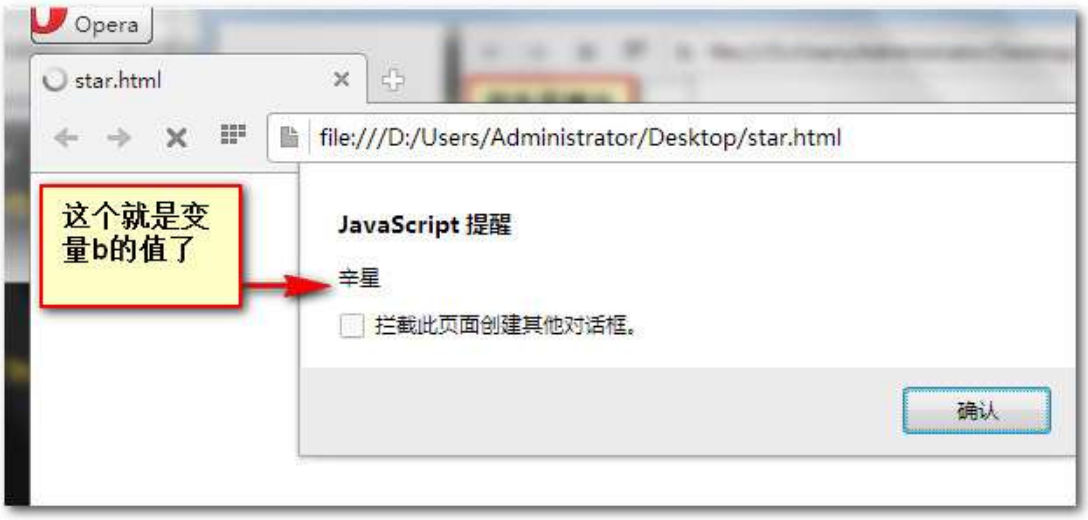
4.我们亲自写一个实例吧，我们在我们的 star.js 中写如下代码：

5.然后 star.html 代码中依旧是这个样子：

6.然后我们运行页面，首先看到变量 a，看下效果吧：



然后我们点击“确定”，之后看到变量 b 的效果图：



7.当然啦，我们这里只是拿其中的两个变量来试了一下。

>>>>>>>>>>运算符 <<<<<<<<<<<<<<<<<<<

1.变量之间是可以运算的，运算符和其他编程语言也基本一样。

2.首先我们看下最简单的逻辑运算符：

运算符	描述
&&	逻辑与
||	逻辑或
!	逻辑非

3.然后我们看下比较运算符：

运算符	描述
==	等于
===	绝对等于(值和类型均相同)
!=	不等于
!==	绝对不等于(值和类型均不相同)
>	大于
<	小于
>=	大于或等于
<=	小于或等于

4.然后就是算术运算符：

运算符	描述
+	加号
-	减号
*	乘号
/	除号
++	自增
--	自减

5.然后就是赋值号：

运算符	原表达式	相当于
=	x = y;	x = y;
+=	x +=y;	x = x+y;
-=	x -= y;	x = x-y;
/=	x /= y;	x = x/y;
%=	x %=y;	x=x%y;

6.两个特殊的运算符：

①对于字符串来说，+号可以起到连接作用。

②三目运算符，a? b：c，如果 a 为 true，那么取值为 b，否则取值为 c。

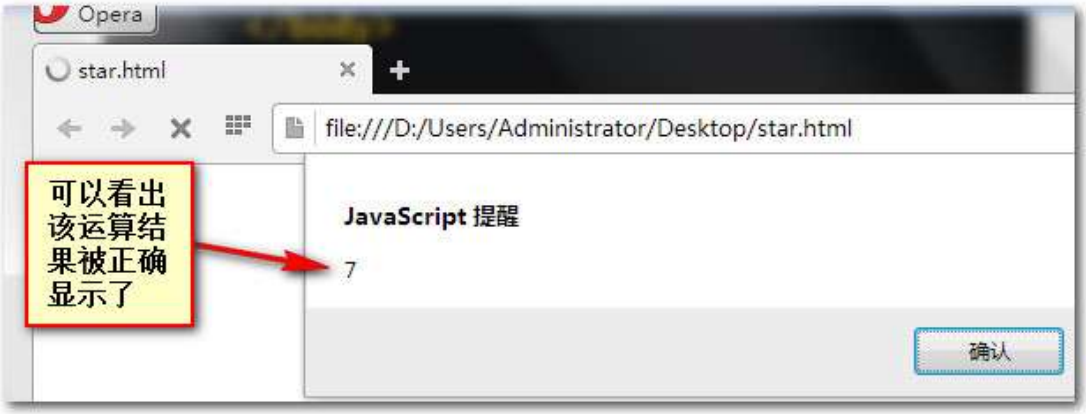
7.那么我们在 star.js 中书写如下代码：

```
var a = 3;
var b = 4;
var c = a + b ;
alert(c);
```

8.然后我们的 star.html 文件还是原来的形式：

```
<html>
    <body>
        <script src="star.js" ></script>
    </body>
</html>
```

9.然后我们用浏览器查看该页面，效果如下：



10.我们可以看出，这里的变量的运算是没问题的。

>>>>>>>>>>小结 <<<<<<<<<<<<<<<<<<<<

1.本小节我们介绍了 Javascript 的变量、数据类型、运算符。

2.由于我默认读者朋友们是学习过其他编程语言的，因此我在这里只介绍了比较重要的定义，在很多细节上都是一笔带过的。

第二节：语句、函数、作用域

>>>>>>>>>>>>>语句<<<<<<<<<<<<<<<<<<<<<<<

1.绝大多数编程语言都有语句的概念，Javascript 也不例外。

2.我们知道在流程控制结构分为三种：①顺序结构②条件结构③循环结构。

3.而这三种结构中的条件结构和循环结构也就对应我们的条件语句和循环语句。

>>>>>>>>>>>条件语句<<<<<<<<<<<<<<<<<<<

1.Javascript 中的条件语句分为四种形式：①if 语句。②if...else 语句③if.....else if....else 语句。④switch 语句。

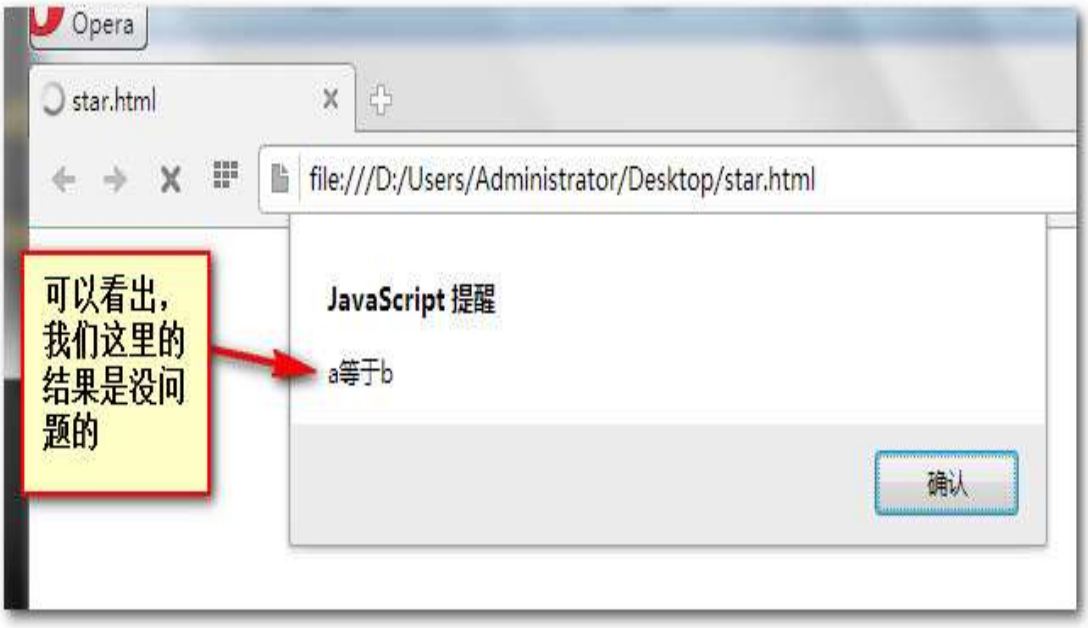
2.这里我们仅以 if.....else if....else 语句为例来介绍一下，我们首先在 star.js 中书写如下代码：

```
var a = 3;
var b = 3;
if(a > b){
    alert("a大于b");
}else if(a == b){
    alert("a等于b");
}else{
    alert("a小于b");
}
```

3. 当然我们在 star.html 中的代码如下：

```
<html>
    <body>
        <script src="star.js" ></script>
    </body>
</html>
```

4.那么它的效果如下：



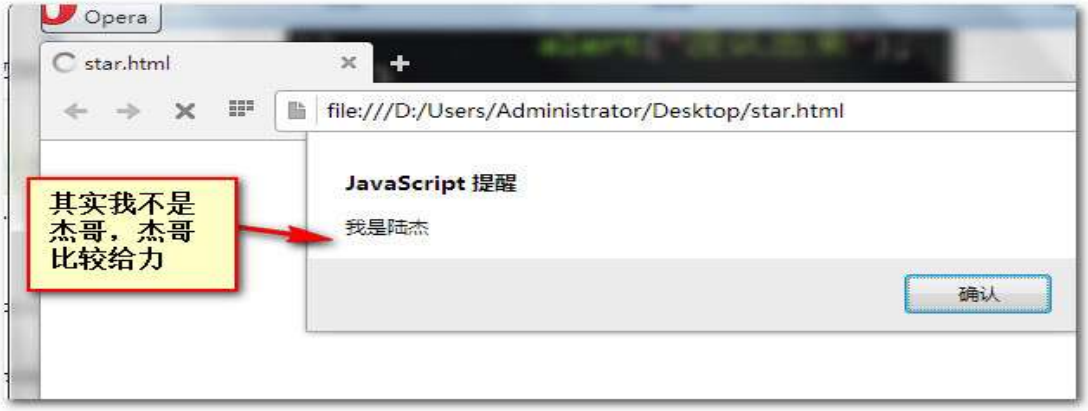
5.然后说说 Javascript 的 switch 语法吧，它的语法格式如下：

```
switch(变量)
{
        case 值 1：
                代码块 1
                break；
        case 值 2：
                代码块 2
                break；
      ..............
        default：
                代码块 n；
}
```

6.那么我们来写一个实例，我们的 star.html 还是不变的，我们的这里还是单纯的修改 star.js 中的代码：

```
var a = 3;
switch(a){
    case 1:
        alert("我是辛星");
        break;
    case 2:
        alert("我是小倩");
        break;
    case 3:
        alert("我是陆杰");
        break;
    default:
        alert("没认出来");
}
```

7.然后我们看下效果图吧：



8.对于条件语句，我们就讲到这里啦。

>>>>>>>>循环语句<<<<<<<<<<<<<<<<<<<<<<<<

1.对于循环语句来说，和大多数语言差不多，支持 for 循环和 while 循环。

2.对于 for 循环，语法也是很类似的：

for(初始条件；是否结束；渐变条件){

.....循环体

}

3.对于 for...in 这种语法，这里就不介绍了，等我们后面用到了数组的时候再介绍。

4.对于 while 循环，其实是有 while 循环和 do...while 循环的，首先是 while 循环的格式：

while(条件){

.....循环体

}

然后就是 do.....while 循环的格式：

do{

.....循环体

}while(条件)；

5.然后我们就 do...while 循环给个例子吧，首先我们在 star.js 中给出下面代码：

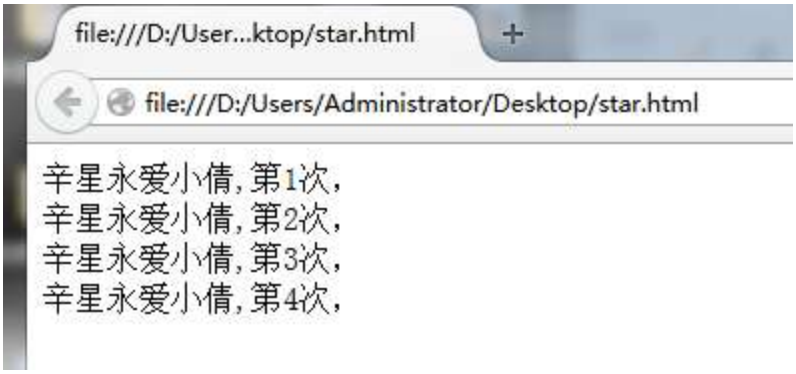
```
var a = 1;
do{
    document.write("辛星永爱小倩,第");
    document.write(a);
    document.write("次, <br />");
    a ++;
}while(a < 5);
```

6.然后我们在 star.html 中写入如下代码：

```
<html>
    <body>
        <script src="star.js" ></script>
    </body>
</html>
```

7.这里说下吧，这个 document.write()；其实到了后面讲 DOM 的时候会讲的，它的意思就是向页面上写入数据的意思。

8.然后我们看看此时的页面效果贴吧：



9.是否和朋友们想的差不多呢?

>>>>>>>break 和 continue<<<<<<<<<<<<<<<<<<

1.首先说 break 吧，它用于跳出循环，而 continue 用于跳过循环的一个迭代。

2.用一句比较通俗的话来说，就是 break 相当于“彻底结束”，而 continue 则是“重头再来”。

3.由于我相信在绝大多数语言中大家都能见到这俩兄弟，因此这里就不过多介绍了。

>>>>>>>>>>>>>函数<<<<<<<<<<<<<<<<<<

1.所谓函数，就是被封装起来的可重复调用的代码块。

2.在 Javascript 中定义函数的格式如下：

```
function  函数名(参数名 1，参数名 2.....，参数名 n){
    .....函数体
}
```

3.这里值得注意的是不管是否有返回值，这个 function 是不要变的，而且 Javascript 的函数中的参数列表中不允许写参数名，只需要写参数值就可以了，如果有返回值，使用 return+返回值这种格式。

4.要说到函数名的命名规则，一般我们采用小驼峰的方式，也就是组成这个函数名的第一个单词小写，之后的每个单词的首字母都大写，比如 getMyNameAndMyAge 就是一个合法的函数名，而且函数名区分大小写奥。

5.那么我们就来定义一个函数吧，我们首先在 star.js 中写如下代码：

```
function xin(a,b){
    return a+b;
}
document.write("最后的结果是:");
document.write(xin(3,4));
```

6.然后 my.html 中的代码为：

```
<html>
    <body>
        <script src="star.js" ></script>
    </body>
</html>
```

7.那么最终我们的网页效果如下：

8.对于函数，我们就了解这些吧，当然，这是基于朋友们有其他语言的功底的情况下。

>>>>>>>>>>>>>变量作用域<<<<<<<<<<<<<<<<<<<

1.我们经常会讨论到变量的作用域和生命周期。

2.在 Javascript 中，我们一般分为全局变量和局部变量。

3.在 Javascript 的函数内部使用 var 声明的变量就是局部变量，只能在函数内部使用。当函数执行完毕之后，局部变量就会被删除。

4.在函数外声明的变量就是全局变量，网页上的所有脚本和函数都可以访问它。当页面关闭之后，全局变量就会被删除。

5.如果我们把值赋予未声明的变量，那么该变量就会自动变成全局变量，就算它是在函数内部。如果 xin 这个变量从未声明，那么当我们如下操作 xin=23；那么该语句会自动给 xin 赋值为 23，而且 xin 这个变量即成为全局变量。

>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<<<

1.由于本书是默认读者有一定的编程语言基础的，因此在介绍这部分的时候走马观花的介绍了一遍。

2.它的作用是让读者朋友们快速认识到 Javascript 语言和其他语言的一些不同之处，而不是让读者朋友们把 Javascript 当做入门语言。

3.说实话，如果您把 Javascript 当做入门语言，我个人是极度排斥的，它本身就像一个大杂烩，不是一个很“纯正”的语言。

第三节：数组、对象、json

>>>>>>>>>>>>>说明<<<<<<<<<<<<<<<<<<<<<<

1.我们这里还是采用之前的 star.html 和 star.js 的方式，而且 star.html 的代码始终如下：

```
<html>
    <body>
        <script src="star.js" ></script>
    </body>
</html>
```

2.我们的输出方式也是采用 document.write(xx)；这种方式来输出内容，即直接输出到页面上。

>>>>>>>>>>>>>数组<<<<<<<<<<<<<<<<<<<<<<

1.数组就是使用一个变量来存储一组数据，由于 Javascript 是弱类型语言，因此这一组数据的数据类型可以相同，也可以不同。

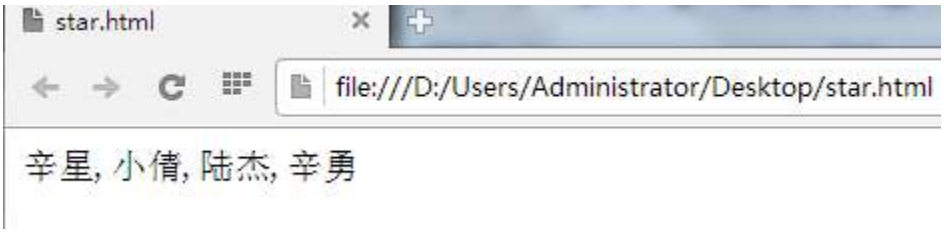
2.对于数组的创建，通常来说有三种方法：

①先创建后赋值（下面是实例代码）：

```
var xin = new Array();
xin[0] = "辛星";
xin[1] = "小倩";
xin[2] = "陆杰";
xin[3] = "辛勇";

document.write(xin);
```

我们先看下最终的运行效果吧：



然后这里我们分析下源代码，这里的 xin 是一个数组名，而且该数组的下标是从 0 开始的，因此我们从 xin[0]开始为数组元素赋值。

②创建的同时赋值(下面是实例代码):

```
var xin = new Array("陆杰","小倩","辛星");
document.write(xin);
```

我们可以看出，这里直接把数组的元素直接在使用 Array 这个类创建的时候就进行了赋值，那么我们看下运行效果吧：

③简写形式(下面是实例代码):

```
var xin=["辛勇","陆杰","辛星","小倩"];
document.write(xin);
```

然后我们这里直接使用中括号来新建一个数组，其中每个元素使用逗号分开。我们看下运行效果吧：

3.说完了数组的创建，那么这里 Javascript 的数组的下标都是从 0 开始的，而且访问其中的某个元素可以使用中括号内写下标的方式。

>>>>>>>>>>面向对象 <<<<<<<<<<<<<<<<<<<<<<<<

1.虽然说绝大多数语言现在都支持面向对象了，但是它们在面向对象的实现方式上的差别，还是蛮大的。

2.这里我们假设读者有面向对象的基础，不管是通过哪门语言接触到的面向对象。

3.在 Javascript 中所有的事物都是对象，包括字符串、数值、数组，甚至是一个函数。

4.而且 Javascript 提供了很多内置对象，其实从简单易用的角度来看，使用这些内置对象就可以解决很多问题了。

5.Javascript 是基于原型的，不是基于类的。

>>>>>>>>>>>>>>对象<<<<<<<<<<<<<<<<<<<<

1.对象是一种特殊的数据，它拥有属性和方法。

2.所谓属性，就是与对象绑定了的变量，所谓方法，就是与对象绑定了的方法。

3.在 Javascript 中创建对象的方式很多，这里介绍其中比较简单的几种方式吧。

4.首先我们看最原始的方法，它是通过首先 new 一个 Object 的对象，然后通过设置它的属性和方法来完成对象的创建，我们看如下代码：

```
var person     = new Object();
person.myname = "辛星";
person.myaim   = "还能再战十年";
person.msg     =  function(){
    document.write("我是"+this.myname+":"+this.myaim);
}
person.msg();
```

5.这种最原始的方式，首先我们看到 person 是一个对象，然后设置其 myname 和 myaim 属性，最后设置了一下 msg 方法，最后我们调用了这个方法。

6.最终它的结果如下：

7.于是就出现了一个工厂方法，所谓工厂方法，就是我们写一个方法，它可以批量的创建对象，这种思想我想朋友们应该可以理解吧，我们首先看如下代码：

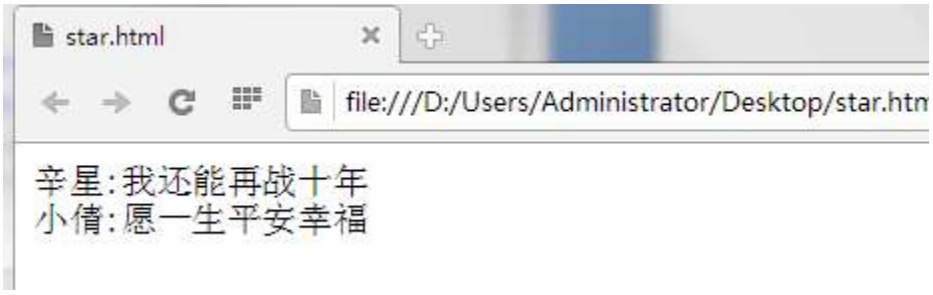
```
function createPerson(myname,myaim){
    var p  = new Object();
    p.myname = myname;
    p.myaim  = myaim;
    p.msg    = function(){
        document.write(this.myname+":"+this.myaim+"<br />");
    }
    //别忘记返回这个值奥
    return p;
}

var xin = createPerson("辛星","我还能再战十年");
xin.msg();
createPerson("小倩","愿一生平安幸福").msg();
```

8.我们可以看到，上面我们写了一个方法 createPerson，该方法接受的参数会被用来设置为对象的属性值，而且它会把创建的对象返回出来。

9.这样，我们就可以调用这个方法来反复的创建对象了，比如下面我们首先创建了一个 xin 这个对象，然后调用了该对象的 msg 方法，当然我们也可以直接使用 createPerson 方法的返回值直接调动 msg 方法。

10.这里咱们还是看下效果吧：



11.其实我们还可以使用构造函数的方法去创建一个对象，比如我们想创建一个 person 对象，它有两个属性 myname 和 myaim，有一个方法 msg，那么我们就可以写一个方法 person，在里面对属性进行赋值，然后给出这个函数的实现。

12.我们看下面代码：

```
//首先我们写一个构造函数
function Person(myname,myaim){
    this.myname = myname;
    this.myaim  = myaim;
    this.msg    = function(){
        document.write(this.myname+":"+this.myaim+"<br />");
    }
}
//创建两个对象
var xin  = new Person("辛星","宝刀不老，再战十年");
var qian = new Person("小倩","生活美满，平平安安");
//分别调用其方法
xin.msg();
qian.msg();
```

13.我们创建了一个 Person 函数，它就是构造函数了，然后它里面使用 this 来给自身的属性和方法进行赋值，然后就是我们创建了两个对象，最后我们就分别调用了它的 msg 方法。

14.我们看下效果图吧：

15.可能有些人会说：既然 Person 这个构造函数中的方法可不可以单独拿出来呢？答案是可以的。不过我们需要在 Person 这个构造函数中指定它的方法，不过此时它不要加括号。

16.我们直接看下示例代码吧：

```
//首先我们写一个构造函数
function Person(myname,myaim){
    this.myname = myname;
    this.myaim  = myaim;
    this.msg    = msg;//这里不要加括号
}
//Person类的msg方法
function msg(){
    document.write(this.myname+":"+this.myaim+"<br />");
}
//创建两个对象
var xin  = new Person("辛星","多少风雨为你遮");
var qian = new Person("小倩","多少良言对你说");
//分别调用其方法
xin.msg();
qian.msg();
```

17.当然它的效果是一样的，我们给结果来个截图吧：

18.首先我们看一下最简单的基于原型的对象产生方式吧。当然由于 Javascript 是基于原型方式的，那么我们就可以把构造函数做成空的，然后把所有的属性和方法直接赋值给 prototype。

19.我们看如下代码：

```
//一个空的构造函数
function Person(){}
//对其原型进行赋值
Person.prototype.myname = "辛星";
Person.prototype.myaim  = "人生历经无数难，背卧沙场风雨旋";
Person.prototype.msg    =  function(){
    document.write(this.myname+":"+this.myaim);
}

var p = new Person();
p.msg();
```

20.那么我们看下效果吧：

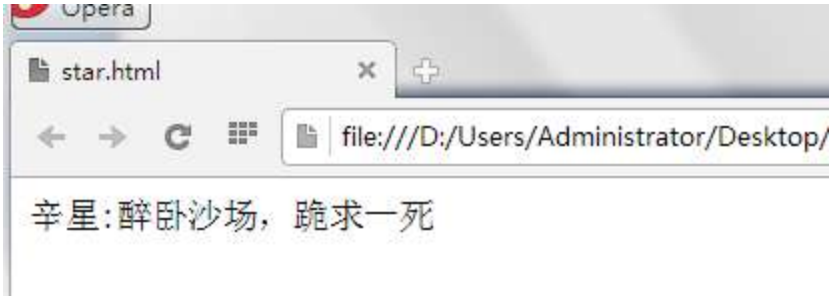
21.可能有人说：你怎么把数据写到原型中了？没错，这么做问题相当多，那么可以用原型+构造函数的方式来创建对象。

22.我们看下示例代码吧：

```
//一个空的构造函数
function Person(myname,myaim){
    this.myname = myname;
    this.myaim  = myaim;
}
Person.prototype.msg    =  function(){
    document.write(this.myname+":"+this.myaim);
}

var p = new Person("辛星","醉卧沙场，跪求一死");
p.msg();
```

23.那么我们看下效果吧：



24.当然啦，我们还有比如动态原型等等很多方法，如果单纯从Javascript 语言的角度来说，确实都必须一一研究透彻。

25.但是如果我们只是简单的想做一个特效，那么我们会几种基本形式就可以了。

>>>>>>>>>>>创建一个对象<<<<<<<<<<<<<<<<<<

1.虽然上面说了一些创建一个对象的方式，但是对于追求简单高效的 Javascript 来说，并不是那么高效。

2.当我们不需要方法的时候，直接使用一个大括号表示一个对象，然后用属性：值作为一个单位，就可以了，大括号中全是这样的单位，我们给个范例：

cat{nickname:“connie”,age:3}

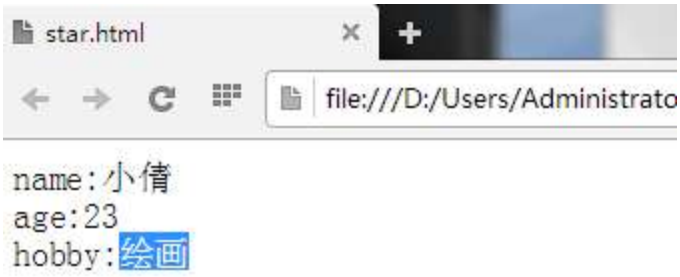
3.还记得我们上面说过的 for....in 吗？下面是它的语法格式：

```
for( 代表变量 in 对象/对象){
        .....其他代码
}
```

4.那么我们先看如下代码吧：

```
//创建一个对象
var qian= {name:"小倩",age:23,hobby:"绘画"};
//分别输出这个对象的属性和值
for( x in qian){
    document.write(x+":"+qian[x]+"<br />");
}
```

5.然后我们看下效果吧：



6.可以看到，这里还是蛮方便的吧。

>>>>>>>>>>>>>json<<<<<<<<<<<<<<<<<<<<<

1.我不知道朋友们是否听说过json的大名，但是毫无疑问它的作用是相当之大的。

2.首先json的英文全称是JavaScript Object Notation，它是一种轻量级的数据交换格式，而且它是独立于语言的，也就是它可以被多种语言去解析。

3.还记得我们的数组、对象的简化表示方式吗？用这种形式去表示数据，就是json的表示方法。

4.首先我们看一段json格式的文本吧：

```
{
    wolf:[
        {name:"辛星",nickname:"小火"},
        {name:"辛勇",nickname:"辛老大"},
        {name:"陆杰",nickname:"定海神针"}
    ]
}
```

5.上面的wolf是一个对象，它的取值是一个数组，而这个数组有三个元素，第一个元素的name是“辛星”，nickname是“小火”，第二个元素的naem是“辛勇”，nickname是.....

6.可以看出json格式还是蛮清晰的，有兴趣的读者可以搜索更多资料，这里暂不展开了。

第二部分：BOM 部分

第零节：说明

>>>>>>>>>>>>>可独立学习<<<<<<<<<<<<<<<<<

1.首先我说一下吧，本部分介绍的 BOM 和前面的基础语法有关系，但是如果读者朋友们接受新知识的速度足够快，也可以直接学习。

2.而且 Javascript 这部分的知识本身就相对独立，因此，我们介绍的时候也就相对分散。

>>>>>>>>>>>>>编写习惯<<<<<<<<<<<<<<<<<<

1.我们这里在同一个目录下新建一个 star.js 文件与 star.html 文件，这里我们来个截图吧：

2.然后我们在 star.html 文件中写入如下内容：

```
<html>
    <body>
        <script src="star.js" ></script>
    </body>
</html>
```

3.之后我们就是修改 star.js 这个文件中的代码了，一般情况下，只要没有特殊说明，代码都是写到 star.js 中的。

4.我们的效果展示使用的浏览器就是 Opera 这款浏览器，当然特殊情况下也会换用别的浏览器。

>>>>>>>>>>>>>给我们建议<<<<<<<<<<<<<<<<<

1.由于该教程并不是很完善，我们继续您的建议来完善它。

2.当然您也可以加入我们，一起来讨论吧。

第一节：DOM 简介

>>>>>>>>>>文档对象模型<<<<<<<<<<<<<<<<<<<<

1.当网页被加载的时候，浏览器会创建一个文档对象模型，英文表述即 Document Object Model，简称为 DOM。

2.我们可以通过 Javascript 来操纵 DOM，我们可以修改页面中的 html 元素、修改页面中的 html 属性，修改相应的样式，并且能够响应页面相应的事件。

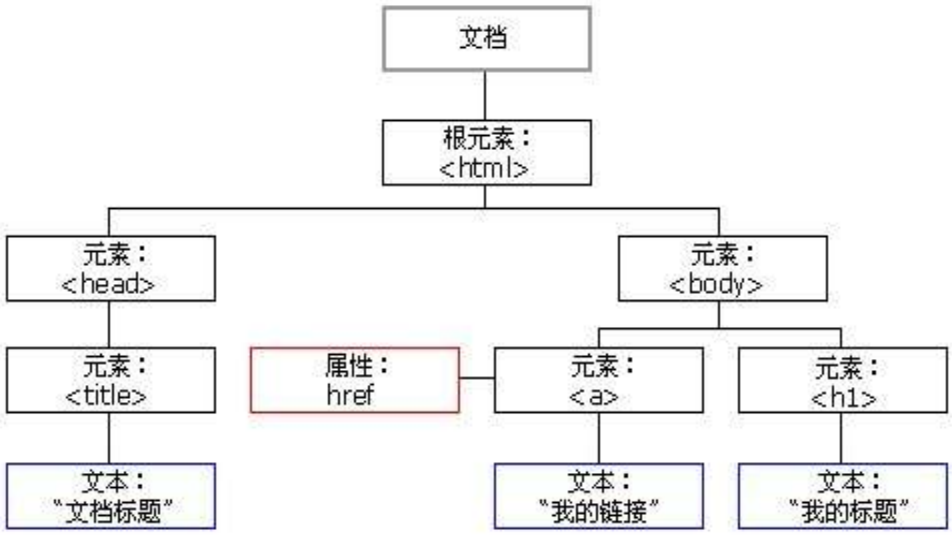
3.我们通过 DOM 可以以编程的方式来修改页面，这点对于 DOM 很重要。

>>>>>>>>DOM 树<<<<<<<<<<<<<<<<<<<<

1.还记得我们网页是由若干元素组成的吗？每个元素可以看做一个节点。

2.那么这些节点以一定的次序排列起来，就形成了一个树状的结构，也就是 DOM 树。

3.下面我们给出一个 DOM 树的典型的图示：



4.可以看出，这里的 html 元素下包含了 head 元素和 body 元素，这种树状结构，不管是之后对于元素的遍历等若干操作，还是我们理解整个体系结构，都有比较大的意义。

5.对于 DOM，我们先介绍这么多。

>>>>>>>操纵元素<<<<<<<<<<<<<<<<<<<<<<<

1.要操纵 html 中的一个元素，我们必须先找到它，我们可以通过如下方式去找，我这里给出一个列表：

途径	使用方法
通过 id	getElementById
通过标签	getElementsByTagName

2.首先我们看下 star.html 中的内容吧：

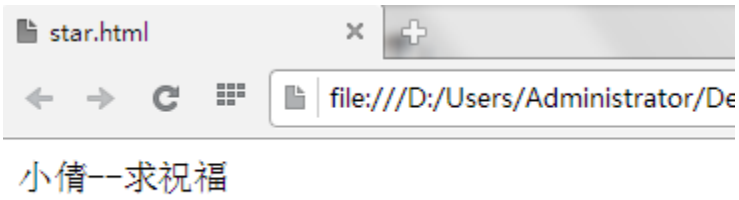
```
<html>
    <body>
        <div id="xin">我是辛星</div>
        <script src="star.js" ></script>
    </body>
</html>
```

3.可以看出它只是定义了一个 div，并且在这个区块中写了“我是辛星”四个字。

4.然后我们可以修改找到的这个 div，然后把它的文本内容改掉，我们在 star.html 中书写如下代码：

```
//首先获得这个元素
var e1 = document.getElementById("xin");
//然后修改这个元素中的文本内容
e1.innerHTML = "小倩--求祝福";
```

5.此时我们刷新页面，看到内容如下：



6.然后我们看到，我们修改页面信息的功能已经完成了。

>>>>>>>>>>>>说明<<<<<<<<<<<<<<<<<<<<<<<<

1.这里我们关键是需要知道它的大致操作流程，知道了它的大致流程，我们做事就方便了很多。

2.它的做事流程大体是这样的：①首先拿到这个元素。②对这个元素进行一系列的操作。

3.这里比较重要的两个方法，一个就是 getElementById，另一个就是 getElementsByTagName,我们上面演示了第一个操作，由于是通过 id 来获取的，这个时候必须保证满足此 id 的元素只有一个，否则就会出现混乱。

4.当然，更多的信息，读者朋友们可以自行搜索奥。

第二节：操纵 html

>>>>>>>>>>说明<<<<<<<<<<<<<<<<<<<<<<

1.通过 DOM，我们可以操纵 html，来完成对元素的查找、属性的修改、删除、增加等操作。

2.这里我们仅仅以比较常见的几种情况进行介绍。

>>>>>>>>>>获取修改 html 文本内容<<<<<<<<<<<<<<

1.这里我们获取和修改 html 文本内容是通过 innerHTML 这个属性来得到的。

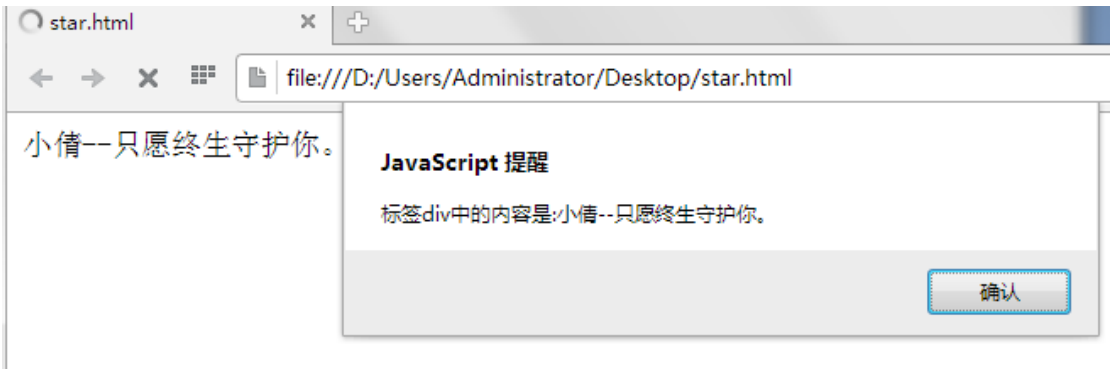
2.比如我们在 star.html 中写入如下代码：

```
<html>
    <body>
        <div id="star">小倩--只愿终生守护你。</div>
        <script src="star.js" ></script>
    </body>
</html>
```

3.然后我们在 star.js 中写入如下内容：

```
var msg = document.getElementById("star").innerHTML;
alert("标签div中的内容是:"+msg);
```

4.然后我们会发现，此时的效果是这样的：



5.上面我们演示的是查看 html 元素中的文本，至于修改文本，由于上一节中讲过了，这里就不再赘述了。

>>>>>>>查看修改 html 元素的属性<<<<<<<<<<<<<<<

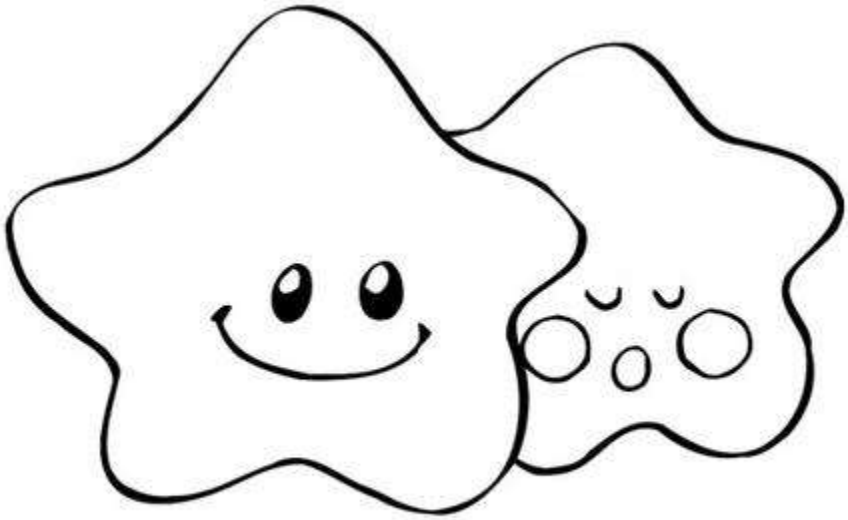
1.我们通过操作相应的个属性即可，通常来说还是两个操作：查看和修改。

2.如果这个属性出现在＝号的左边，那么就是查看它的值啦，如果这个属性出现在＝号的右边，那么就是给它赋值啦。

3.比如说我们在该目录下有两张图片：

①第一张是 cat.jpg，看图如下：

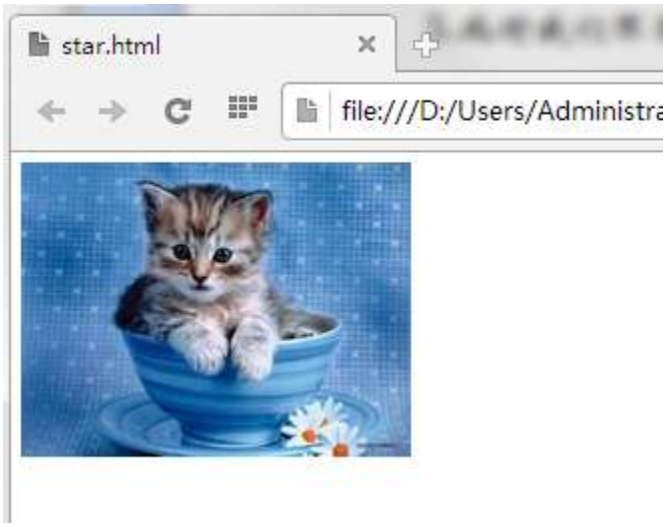
②第二张是 star.jpg，看图如下：



4.至于图片是否漂亮，这个不在本话题讨论范围之内，我们首先看一下 star.html 的代码吧：

```
<html>
    <body>
        <img src="cat.jpg" />
        <script src="star.js" ></script>
    </body>
</html>
```

5.此时我们不写 star.js 文件，那么效果图很明显是这样的：



6.那么我们就来修改下 star.js 吧，它修改了 img 这个标签的 src 属性，看我代码如下：

```
//这里是通过标签来获取的，注意取第一个奥
var e1 = document.getElementsByTagName("img")[0];
//直接给其src属性赋值即可
e1.src = "star.jpg";
```

7.此时效果如下：

8.读者朋友们是否也学会使用了呢？

>>>>>>>输出流<<<<<<<<<<<<<<<<<<<<<<<

1.其实我们之前一直用 document.write，那么它的意思是什么呢？

2.它是在向 DOM 中继续写一些东西，我们之前是向 html 网页中写入信息，注意它是继续向后写的，而不是从头开始写。

3.但是注意一点，就是不要在整个 DOM 都加载完毕之后再去使用它，可能我们目前还不太能够理解，等我们学习了事件之后，我再帮助朋友们解决这个问题。

>>>>>>>>>>>>元素的删除<<<<<<<<<<<<<<<<<

1.我们用 removeChild 来删除父元素中的一个子元素，使用格式：

父元素.removeChild(子元素)；

2.我们首先来看下 star.html 中的内容吧：

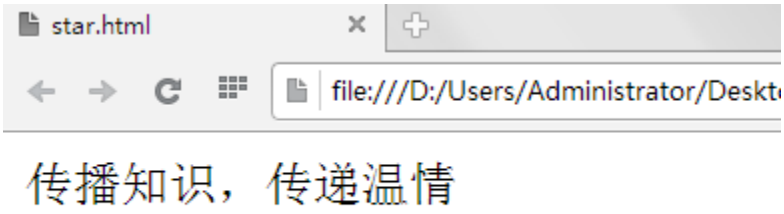
```
<html>
    <body>
        <div id = "xin">
            <p id="xin1">传播知识，传递温情</p>
            <p id="xin2">从横驰骋，我是辛星</p>
        </div>
        <script src="star.js"></script>
    </body>
</html>
```

3.然后我们看下当 star.js 什么都没有的时候的效果吧：

4.那么此时我们就把第 2 个 p 元素给干掉吧，那么我们应该怎么做呢？看我在 star.js 中的代码吧：

```
//获取父元素
var e1 = document.getElementById("xin");
//获取子元素
var e2 = document.getElementById("xin2");
//移除子元素
e1.removeChild(e2);
```

5.然后我们看一下此时的页面效果吧：



6.读者朋友们是否明白了怎么操作呢？

>>>>>>>>>>添加元素<<<<<<<<<<<<<<<<<<

1.要说到添加元素，稍微复杂一些，因为我们一般使用 createElement 来创建一个元素。

2.然后我们使用 父元素.appendChild(子元素)； 的方式来添加一个子元素。

3.我们首先看 star.html 中的内容吧：

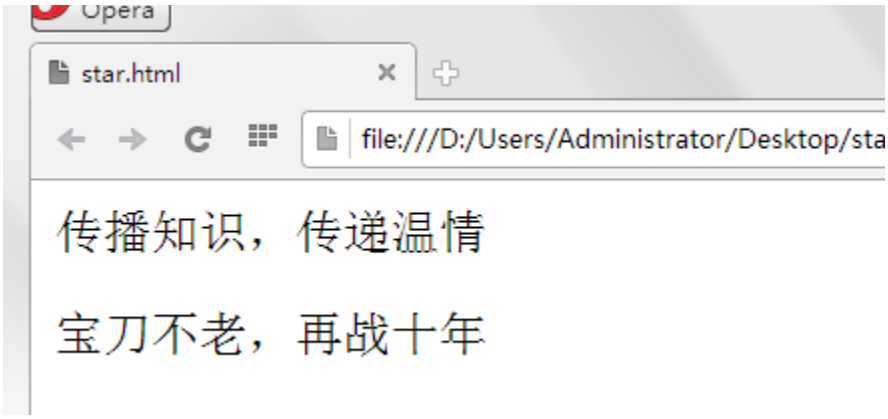
```
<html>
    <body>
        <div id = "xin">
            <p id="xin1">传播知识，传递温情</p>
        </div>
        <script src="star.js"></script>
    </body>
</html>
```

4.如果我们不在 star.js 中写入任何代码，此时效果为：

5.那么我们在 star.js 中写入如下代码：

```
//创建一个p标签
var e1 = document.createElement("p");
//设置它的文本内容
e1.innerHTML = "宝刀不老，再战十年";
//得到它的父元素
var e2  = document.getElementById("xin");
//把子元素添加到父元素中
e2.appendChild(e1);
```

6.此时页面效果如下：



7.是不是很好理解呢？

>>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<<<<

1.本小节我们主要介绍的就是修改 html 元素。

2.主要就是修改 html 文本的内容，修改它的属性等等。

3.当然还有 html 元素的增加和删除，说实话，这个感觉倒不算太常用。

第二节： 操纵 css

>>>>>>>>>>>>修改样式<<<<<<<<<<<<<<<<<<<<

1.由于我们对 css 还是蛮熟悉的，我们看一下如何去修改它的样式。

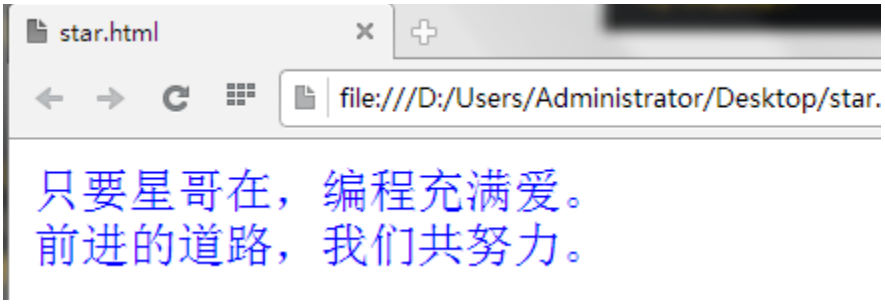
2.首先我们先得到某一个元素，然后格式如下：

元素.style.属性＝值；

3.我们先看 star.html 文件的代码：

```
<html>
    <head>
        <style type="text/css">
            #xin{
                color: blue;
            }
        </style>
    </head>
    <body>
        <p id="xin">
            只要星哥在，编程充满爱。<br />
            前进的道路，我们共努力。
        </p>
        <script src="star.js"></script>
    </body>
</html>
```

4.如果 star.js 中没有内容，那么此时的效果图如下：



5.那么下面我们在 star.js 中添加这么一句话：

```
//设置字体颜色为红色
document.getElementById("xin").style.color="red";
```

6.此时我们再看看效果，发现效果如下：

7.这里我们成功修改了它的样式。

>>>>>>>>>>说明<<<<<<<<<<<<<<<<<<<<<<<<

1.对于修改样式来说，设计到的 Javascript 的东西并不多，更多的是涉及到的 css 的知识。

2.但是很多时候这个功能还是蛮重要的，特别是接下来一节中我们讲到“事件”的时候，重要性就更能体现出来了。

第三节：事件

>>>>>>>>>>>>>说明<<<<<<<<<<<<<<<<<<<<<<

1.记得之前在学习 Win 下的编程的时候，经常会听到一句话“Windows 是基于事件的”。

2.这里的“事件”，简单的理解，就是用户的一些操作，比如用户点击了网页中的某个部分。

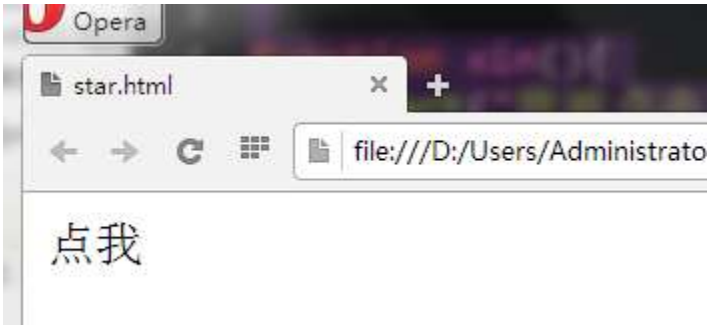
3.我们首先来看个范例，我们在 star.html 中写如下代码：

```
<html>
    <body>
        <p onclick=xin()>
            点我
        </p>
        <script src="star.js"></script>
    </body>
</html>
```

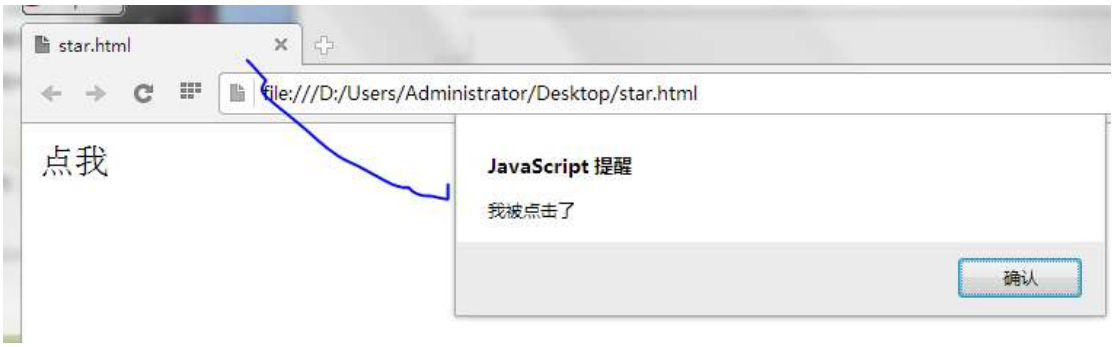
4.然后我们在 star.js 中写如下代码：

```
function xin(){
    alert("我被点击了");
}
```

5.那么我们刚打开网页，效果如下：



6.如果我们点击那个“点我”两个字，就会弹出一个窗口，效果图如下：

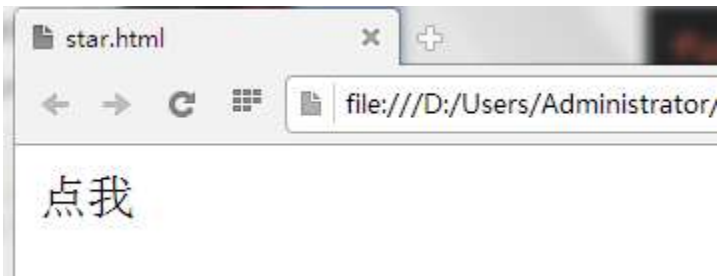


7.这里我解释下源 html 代码吧，上述代码中的 onclick=xin() 表示当这个元素被点击的时候，它会自动调用 xin 函数，而 xin 函数的作用就是弹出一个对话框。

8.还记得我们前面说过的在页面加载完成后使用 document.write() 会让页面原有内容消失吗？我这里给大家演示下，我们直接把 star.js 中的代码改成如下：

```
function xin(){
    document.write("耐住寂寞，刻苦钻研");
}
```

9.此时我们看下页面效果，当然此时页面和刚才是一样的：



10.但是当我们点击了“点我”这两个字之后，接下来的情况就是这样的了：

11.此时的我们对事件点击应该有了一个初步的认识。

>>>>>>>>>>>其他事件<<<<<<<<<<<<<<<<<<<

1.那么主要的事件都有哪些呢?

2.我这里总结了一个列表:

属性	对应事件
onclick	鼠标点击事件
onload	用户进入页面
onunload	用户离开页面
onchange	当内容发生变化时
onmouseover	当鼠标移入时
onmouseout	当鼠标移开时
onmousedown	当鼠标按键按下时
onmouseup	当鼠标按键松开时
onclick	当完成一次点击时
onfocus	当获取鼠标焦点时

3.其实一般我们使用的时候就是对相应的属性进行赋值就可以了,比如 onload = xin()这种格式,来指定当某个事件发生的时候,它会自动调用某个函数,这类函数有个名称,叫做“回调函数”。

4.对于事件的相应,上面已经给出相应的例子了,这里就不再赘述了。

>>>>>>>>>>>事件监听<<<<<<<<<<<<<<<<<<<

1.上面我们的例子中,我们直接把对事件的监听放到 html 代码中了,这样不利于 html 与 js 的分离。

2.如果要做到 html 与 js 的分离,那么 js 代码中必须有为 html 的某些元素添加事件监控的能力,也就是所谓的监听事件。

3.监听事件我们使用 addEventListener 这个方法,它的使用格式如下:

元素.addEventListener(事件,事件处理函数);

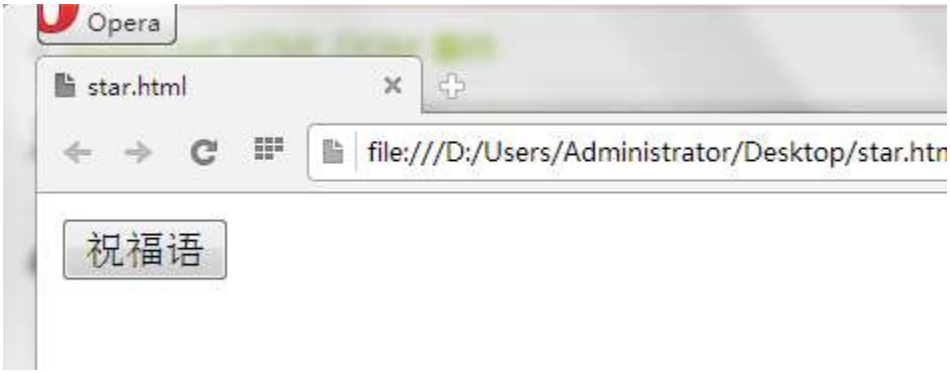
4.说了这么多,我们到练兵场上开始实战演练一下吧,我们首先在 star.html 中书写如下代码:

```html
<html>
    <body>
        <button id="xin">祝福语</button>
        <script src="star.js"></script>
    </body>
</html>
```

5.然后我们在 star.js 中添加如下代码：

```
//定义事件处理函数
function lucky(){
    alert("祝您事事顺心，万事如意");
}
//为该按钮的点击事件添加事件处理函数
document.getElementById("xin").addEventListener("click",lucky);
```

6.那么它的初始界面如下：



7.当我们点击该按钮的时候，效果图如下：

8.可以看出，我们并没有在 html 文件中添加事件监听的代码。

>>>>>>>>>>>关于 addEventListener<<<<<<<<<<<

1.对于这个方法，我们可以向同一个元素添加多个事件句柄，比如即监听 click 事件，又监听 mouseover 事件。

2.其实我们还可以向同一个元素添加多个相同类型的事件句柄，比如我们对一个元素添加两个 click 事件的句柄。

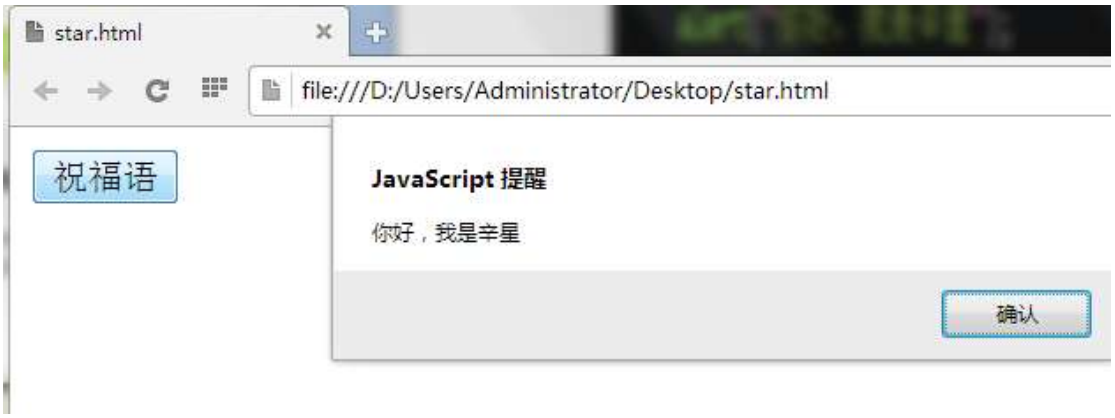
3.我们使用这个方法的一点好处就是：把 Javascript 代码从 html 代码中分离的更加彻底，可读性更强。

4.不过值得注意的是，该方法的第一个参数是没有“on”这个前缀的，而且它的第二个参数是不要加括号的，直接写函数名就可以了。

5.当然，该方法的第二个参数也不一定非得是一个函数名，也可以是一个匿名函数，比如我们还是保持上面的 star.html 代码不变，但是把 star.js 改为如下代码：

```
document.getElementById("xin").addEventListener("click",function(){
    alert("你好，我是辛星");
});
```

6.那么当我们在点击该按钮的时候，就会出现如下的结果了：



7.而且我们也可以利用这一点来实现传递函数的参数，比如如下格式：

```
元素.addEventListener(事件，function(){
        callbcakFunction(参数列表)
}
```

>>>>>>>>>冒泡、捕获<<<<<<<<<<<<<

1.如果朋友们有过界面编程经验，那么会知道事件的捕获是有一定的顺序关系的。

2.比如说我一个 div 元素包含了一个 p 元素，而且这两个事件都会监听 click 事件，那么当我点击 p 元素中的内容的时候，哪一个先触发呢？

3.如果按照我们一般人的逻辑，先触发内部事件比较合理一些，确实，这也是 Javascript 默认的执行方式，但是，很少有事情是绝对的，很多时候我们也会想让外部的事件监听函数先得到响应。

4.这里就引入了 addEventListener 的第三个参数，它的第三个参数是一个布尔类型，规定了事件传递的顺序。

5.下面就是第三个参数的取值列表：

取值	事件传递方式	说明
false(默认)	冒泡	内部元素的事件会首先被触发
true	捕获	外部元素的时间会首先被触发

6.这里我们在 star.html 中书写如下内容：

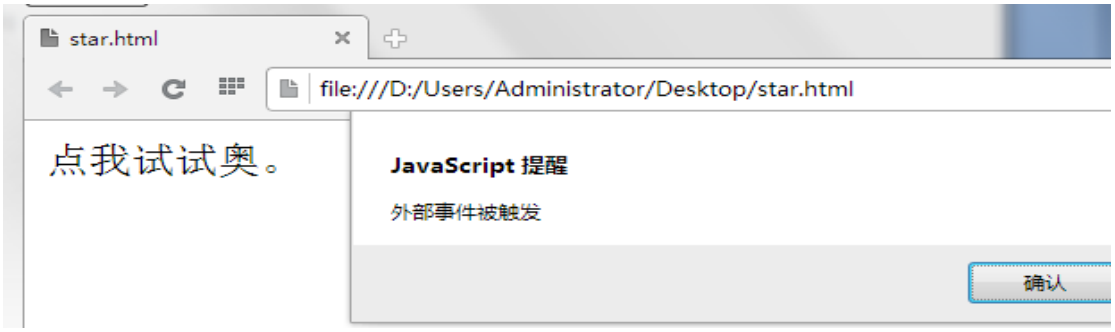
```
<html>
    <body>
        <div id="star">
            <p id="xin">
                点我试试奥。
            </p>
        </div>
        <script src="star.js"></script>
    </body>
</html>
```

7.然后在 star.js 中书写如下内容：

```
//定义一个外部事件
function outter(){
    alert("外部事件被触发");
}
//定义一个内部事件
function inner(){
    alert("内部事件被触发");
}
//分别监听xin和star这两个元素
document.getElementById("xin").addEventListener("click",inner);
//这里强制性设置为先触发外部元素被点击的事件
document.getElementById("star").addEventListener("click",outter,true);
```

8.这里我们强制性的把监听事件的相应方式设置为“捕获”的方式，我们首先看下效果图：

9.然后我们点击该文字，发现首先触发的是 div 这个元素的事件，看到如下效果：



10.然后我们点击“确定”之后，看到如下效果：

11.此时我们就看到它果然是首先相应外部时间，然后响应内部事件。

>>>>>>>>>>>其他知识<<<<<<<<<<<<<<<<<

1.我们使用 addEventListener 来添加事件监听，那么当我们想移除已经监听的事件又该怎么办呢？

2.我们可以使用 removeEventListener 这个方法来完成，它的使用格式如下：

元素.removeEventListener(事件，方法名)；

3.在浏览器兼容这一块，有一个问题始终让我们很头疼，就是 IE 的低版本浏览器，对于 IE8 及其以下的浏览器，我们通常用如下方式来解决：

元素.attachEvent(事件，事件处理函数)；----》添加监听事件

元素.detachEvent(事件，事件处理函数)；----》移除监听事件

>>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<

1.本节中介绍到的主要有事件的分类与事件的监听。

2.之后就是 addEventListener 这个函数的使用。

3.在之后就是“冒泡”与“捕获”这两种事件传递方式，之后就是在低版本的 IE 下的事件的处理，还有就是移除事件的监听。

第三部分：BOM 部分

第零节：说明

>>>>>>>>>>>>>可独立观看<<<<<<<<<<<<<<<<<

1.由于 BOM 和 DOM 部分相互之间还有些交集，因此两者耦合较深。

2.但是 BOM 和 DOM 与前面的基础语法之间的耦合程度其实很低，因此，您可以独立观看本部分。

>>>>>>>>>>>>>>>>编写习惯<<<<<<<<<<<<<<<<<

1.首先我们在同一个目录下新建两个文件，一个是 star.js 用来书写 Javascript 代码，另一个是 star.html 用来书写 html 代码，我们来个截图吧：

2.然后我们在 star.html 中书写如下内容，并且如无特殊说明，我们是不会变该文件的：

```
<html>
    <body>
        <script src="star.js" ></script>
    </body>
</html>
```

3.然后我们剩余其他节的代码都是直接在 star.js 中写的，我这里给朋友们交代清楚之后，后面就不会反复提及了。

4.如果您感觉这种方式有问题，不妨及时和我们取得联系，在后续版本中肯定会得到进一步的修正的。

第一节：BOM 简介

>>>>>>>>>>浏览器对象模型<<<<<<<<<<<<<<<<<<

1.所谓 BOM 就是“Browser Object Model”，也就是浏览器对象模型。

2.我们可以使用 BOM 提供的一些方法和属性来控制浏览器的行为。

3.所有的浏览器中都支持一个 window 对象，它表示一个浏览器窗口。

4.所有的 Javascript 全局对象、函数、变量都自动成为 window 对象的成员。

>>>>>>>>>>>>>.window <<<<<<<<<<<<<

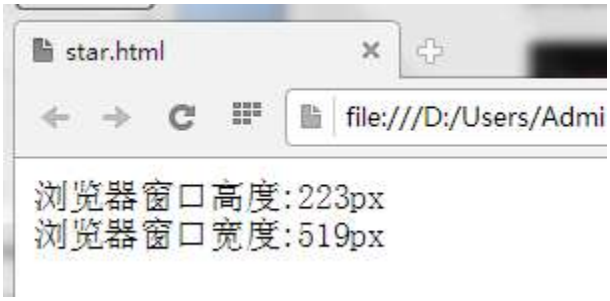
1.对于 window 的设置，其实就是对浏览器窗口的设置。

2.我们可以用 innerHeight 来得到浏览器窗口的内部高度，用 innerWidth 来得到浏览器窗口的内部宽度。

3.比如我们看如下代码：

```
var h = window.innerHeight;
var w = window.innerWidth;
document.write("浏览器窗口高度:"+h+"px");
document.write("<br />");
document.write("浏览器窗口宽度:"+w+"px");
```

4.那么它的运行效果如下：



5.这里是对浏览器窗口大小的测试，不过一般来说对我们的意义并不大。

>>>>>>>>>>>screen<<<<<<<<<<<<<<<<<<<<<

1.上面我们介绍了 window 对象，那么接下来我们认识一下 screen 对象吧。

2.所谓 screen 对象，其实是 window.screen，但是通常在使用的时候省略 window 前缀。

3.通常来说 screen 这个对象的两个属性如下：

属性	含义
availWidth	可用的屏幕宽度
availHeight	可用的屏幕高度

4.比如我们在 star.html 中写如下内容：

```
<html>
    <body>
        <script>
            document.write("屏幕的可用宽度"+screen.availWidth+"px");
            document.write("<br />");
            document.write("屏幕的可用高度"+screen.availHeight+"px");
        </script>
    </body>
</html>
```

5.那么它的输出结果是：

6.注意这里是我们电脑屏幕的分辨率，而且是除去任务栏的分辨率，就是说，我们电脑桌面上的屏幕，除去底层(一般是底层)任务栏之后所剩的分辨率，看我截图：

7.要想得到这个截图，可以在桌面上点右键，然后选择“屏幕分辨率”来得到。

>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<<<<<<

1.本节主要理解什么是 BOM 就可以了。

2.而且本节讲到的是 window 这一个对象，但是我们通常并不用调整它。

3.因此，对于其他的一些方法，我这里就暂不介绍了。

4.那么什么时候通常对 window 和 screen 感兴趣呢？就是在我们考虑到用户体验的时候，根据用户屏幕的大小来适当的调整浏览器窗口的大小，不过一般来说也并不是那么常用，因为更多的时候是采用的是默认值。

第二节：计时、cookie

>>>>>>>>>>>>>>计时<<<<<<<<<<<<<<<<<<<<<

1.有所我们需要对时间进行控制，比如一个考试系统需要记录考生的答题时间，那么就用到 Javascript 的计时功能了。

2.在 Javascript 中使用计时功能是蛮简单的，这里有两个比较重要的方法，下面是它们的列表：

方法	作用
setInterval	每隔 xx 毫秒就去执行指定代码
setTimeout	xx 毫秒之后去执行指定代码
clearInterval	停止 setInterval 方法执行的代码
clearTimeout	停止 setTimeout 方法执行的代码

3.既然有了理论基础，那么我们就来实际操作一下吧。

>>>>>>>>>>>>>代码实战<<<<<<<<<<<<<<<<<<<<<

1.既然有了上述基础，那么我们就来做一个计时器吧，首先我们在 star.html 中写入如下代码：

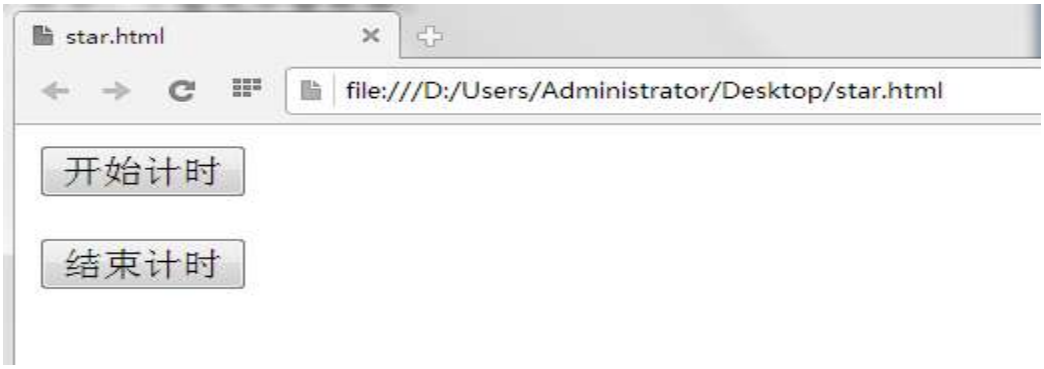
```
<html>
    <body>
        <button id="begin">当前时间</button>
        <p id="time"></p>
        <button id="stop">结束计时</button>
        <script src="star.js"></script>
    </body>
</html>
```

2.我们的目的是点击“当前时间”的时候就开始计时，当点击“结束计时”的时候，就暂停计时，那么该怎么完成呢？

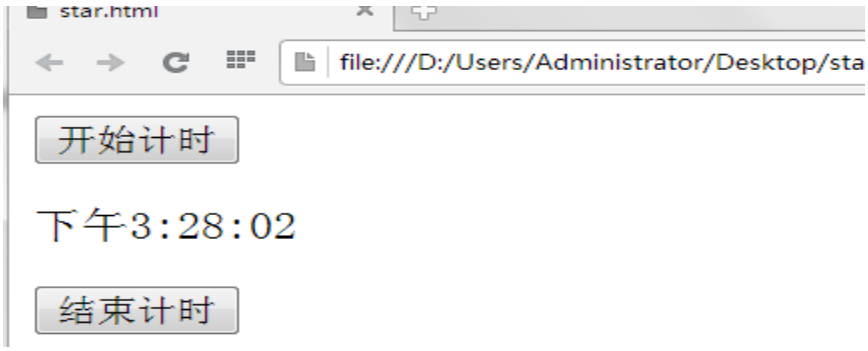
3.首先我们必须得设置一下时间的监听、然后用到开启计时功能、关闭计时功能等的代码，我们在 star.js 中书写代码如下：

```
//得到时间
function myTime(){
    var t  = new Date();
    var mt = t.toLocaleTimeString();
    document.getElementById("time").innerHTML=mt;
}
//开始计时
function start(){
    timeLine = setInterval(myTime,1000);
}
//终止计时
function stop(){
    clearInterval(timeLine);
}
//事件绑定
document.getElementById("begin").addEventListener("click",start);
document.getElementById("stop").addEventListener("click",stop);
```

4.那么我们可以看下效果，下面是界面效果：



5.当我们点击“开始计时”之后，就会每秒钟刷新一次去计时，看下面效果图：



6.当然，当我们点击“结束计时”的时候，计时就会结束了，这一点，我想朋友们可以自己去试试。

>>>>>>>>>>>>>说明 <<<<<<<<<<<<<<<<<<<<

1.对于 timeout 系列的函数，只会执行一次，当然，我们也可以在它没发生之前给 clear 掉。

2.对于 interval 系列的函数，使用会更多一些，多用于周期性的做某些事。

>>>>>>>>cookie<<<<<<<<<<<<<<<<<<<<<<

1.通常 cookie 是一些数据，它存储在我们电脑上的文本文件中。

2.我们知道 web 服务器向浏览器发送 web 页面的时候，是无法记录用户状态的，而 cookie 的作用就是负责“记录用户的某些信息”。

3.而 cookie 以键值对的方式保存，比如“age=23”这种方式。

4.Javascript 可以使用 document.cookie 这个属性来创建、读取、删除 cookie。

5.首先是创建或者修改一个 cookie，操作格式如下：

```
document.cookie = “键=值”;
```

6.当我们要读取 cookie 时，可以使用如下形式(它会以字符串的形式返回所有的 cookie 值)：

```
var x = document.cookie;
```

7.在创建 cookie 的时候还可以添加诸如过期时间、路径等信息，这里给了一个范例代码：

```
document.cookie= “name=xin;expire = Mon, 20 Dec 2014 15:47:00 GMT;path=/”;
```

8.要删除一个 cookie 特别简单，只需要把 expires 参数设置为之前的时间即可，此时我们不必指定 cookie 的值。

>>>>>>>>>>>实战演练<<<<<<<<<<<<<<<<<<<

1.既然我们已经有了上面的知识储备，那么就开始进行实际的操作一下吧。

2.不过值得注意的是我们必须用客户端来访问浏览器的方式去使用才可以，如果没有开启服务器，仅仅是在本地写了 html 和 js 文件，去实验的时候是不会成功的。

3.我们这里使用的依然是 wamp，当然这是 php 语言的集成开发环境，如果您的后台使用的是其他语言，那么就另当别论了。

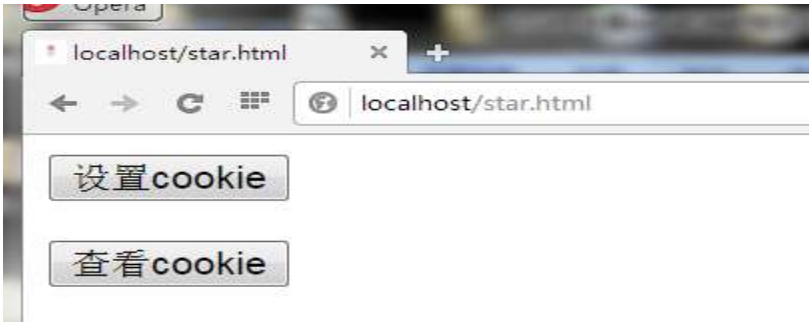
4.我们在根路径下写一个 star.html 文件，内容如下：

```
<html>
    <body>
        <button id="begin">设置cookie</button>
        <p id="cookie"></p>
        <button id="stop">查看cookie</button>
        <script src="star.js"></script>
    </body>
</html>
```

5.然后在同级目录下写一个 star.js 文件，内容如下：

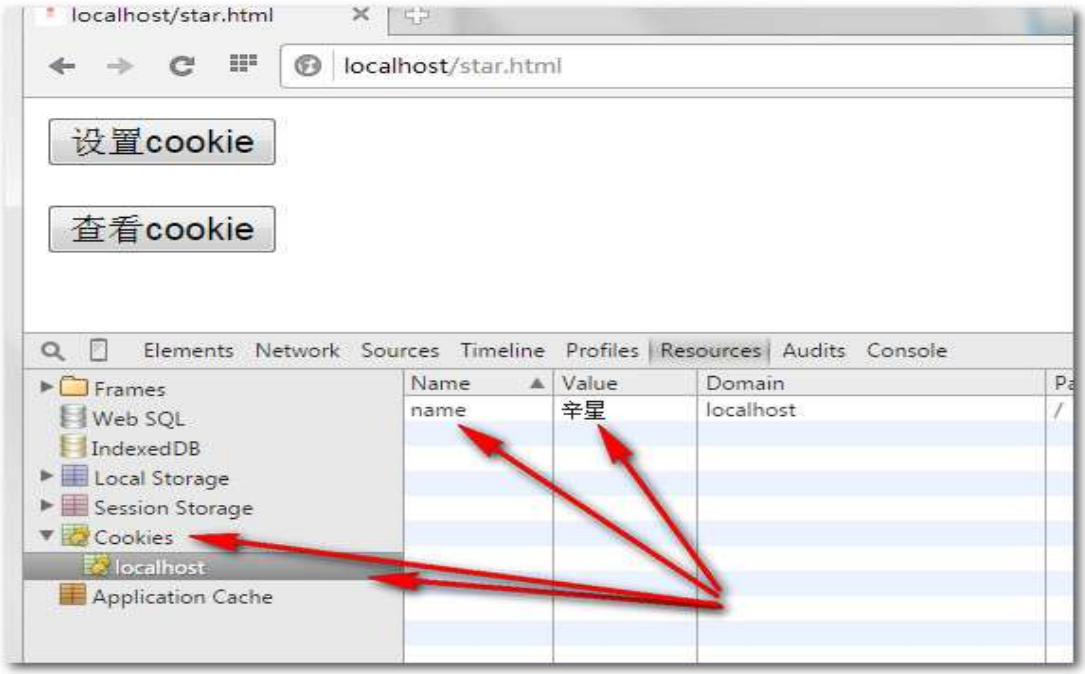
```
//设置cookie
function start(){
    document.cookie="name=辛星";
}
//查看cookie
function stop(){
    var text = document.cookie;
    alert(text);
}
//事件绑定
document.getElementById("begin").addEventListener("click",start);
document.getElementById("stop").addEventListener("click",stop);
```

6.对于这个 html 文件和 js 文件都是什么意思，我想不用我多说了吧，可能朋友们也已经知道是什么意思了。那么我们来看下效果吧：



7.我们首先点击“设置 cookie”，然后点击“查看 cookie”，就会发现它的效果图如下：

8.然后我们看到我们的 cookie 被成功的设置了，当然了呢，我们在下面的 web 检查器中看看效果吧：



9.当然了，使用 Javascript 的方式来操纵 cookie，在一定程度上提供了方便，不过它并不是很安全，也就是它存储的数据是会被我们看到的，而且我们下载一些浏览器插件就可以修改它了，但是，不得不说 cookie 还是有着比较强大的实际意义的。

>>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<<

1.本小节介绍了两个还算常用的知识点。

2.一个就是计时系统，这个要说常用吧，也不是那么常用，要说不常用吧，也有不少时候需要用。

3.第二个就是 cookie，由于我们服务器端的语言也是可以设置 cookie 的，而且一般都是加密之后的 cookie，因此，这两者很多时候需要结合起来使用。

第三节：location、history

>>>>>>>>>>location<<<<<<<<<<<<<<<<<<<<<<<

1.首先说下 location 对象，它用于管理当前页面的地址，也就是 url。

2.下面我们来看一下它的几个属性,下面给出列表：

属性名	含义
hostname	web 主机的域名
pathname	当前页面的路径和文件名
port	web 主机的端口
protocol	使用的 web 协议
href	整个 url

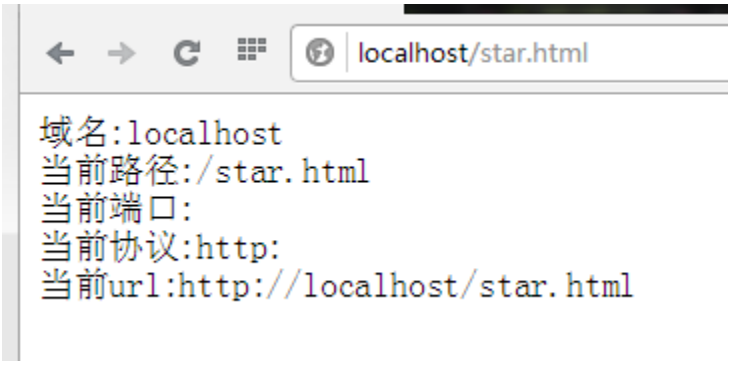
3.然后我们在 star.html 中写入如下内容：

```
<html>
    <body>
        <script src="star.js"></script>
    </body>
</html>
```

4.然后我们在 star.js 中写入如下内容：

```
document.write("域名:"+location.hostname+"<br />");
document.write("当前路径:"+location.pathname+"<br />");
document.write("当前端口:"+location.port+"<br />");
document.write("当前协议:"+location.protocol+"<br />");
document.write("当前url:"+location.href);
```

5.那么我们在浏览器中输入 localhost/star.html 来访问该页面时，截图如下：



6.这里的端口没有写，默认就是 80 端口。

>>>>>>>>>>>页面跳转<<<<<<<<<<<<<<<<<<<<

1.要说到页面跳转，有两种方法：①第一种方法就是 location.href，给这个属性赋值。②使用 location.assign(域名)；这个方法来进行跳转。

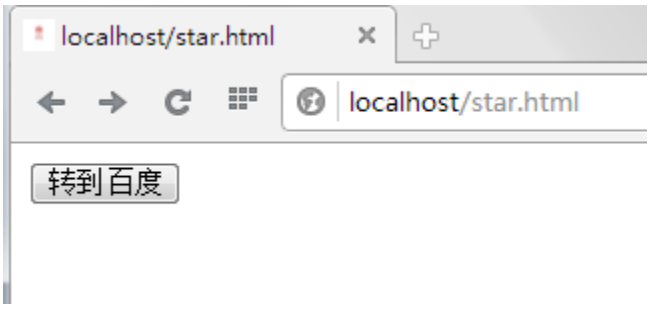
2.我们的 star.html 代码如下：

```
<html>
    <body>
        <button onclick=xin()>转到百度</button>
        <script src="star.js"></script>
    </body>
</html>
```

3.然后我们在 star.js 中写入如下代码：

```
function xin(){
    location.href="http://www.baidu.com";
}
```

4.然后我们看到效果如下：



5.当我们点击“转到百度”这个按钮的时候，他就会自动跳转到百度页面了：

6.上面我们是通过修改 location 的 href 属性来达到网页跳转的目的，接下来我们就通过 location 的 assign 这个方法来做到吧，我们保持上面的 star.html 文件不变，然后我们修改它的 star.js 文件如下：

```
function xin(){
    //通过assign方法来到达百度首页
    location.assign("http://www.baidu.com");
}
```

7.这里效果是一样一样的，具体我就不给朋友们演示了。

>>>>>>>>>>history<<<<<<<<<<<<<<<<<<<<<<<<

1.这个对象表示浏览器的历史，它主要就是两个方法，我们给出如下列表：

方法	描述
history.back()	相当于浏览器中的“后退”
history.forward()	相当于浏览器中的“前进”

2.由于这两个功能实在是比较简单，我还是简单演示下吧，首先我们在 star.html 中书写如下内容：

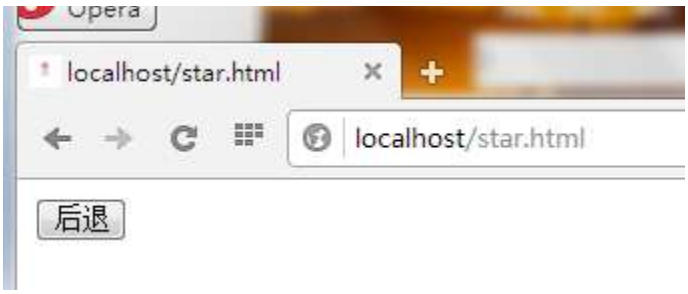
```
<html>
    <body>
        <button onclick=xin()>后退</button>
        <script src="star.js"></script>
    </body>
</html>
```

3.然后我们在 star.js 中写入如下内容：

```
function xin(){
    history.back();
}
```

4.然后我们首先在浏览器中输入 www.baidu.com 来浏览器下百度的页面：

5.然后我们在浏览器中输入 localhost/star.html 来看下效果：



6.然后我们点击“后退”按钮，看下效果吧：

7.上面我们讲解了 history.back(),对于 history.forward()想必朋友们也应该差不多了解了吧。

8.那么什么时候用到这两个属性呢？其实吧，多数是自动跳转的时候会用到，就是比如说我们登录成功，自动跳转到首页，这种情况下，它的使用还是蛮频繁的。

>>>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<<

1.可能本小节的任务是所有任务里面最轻松的。

2.我们首先讲了一下 location 是什么意思，它就是表示我们浏览器的地址栏的内容。

3.然后就是这个 history 这个属性，它就是表示我们的浏览历史，常用的其实就是 history.bcak(),比如我们提交的信息不合法，或者是某个步骤操作失败，就可以使用 history.back()来返回到上一步继续操作，清楚吧。

第四节：总结

>>>>>>>>>>>>>说明<<<<<<<<<<<<<<<<<<<<

1.其实 Javascript 有点像我们高中学习的化学，知识点比较零散。

2.这些知识点不学吧，没安全感，学吧，又很零散。

>>>>>>>>>>>>弹出窗口<<<<<<<<<<<<<<<<<<<<

1.到现在我们比较常用的弹出窗口的方式就是 alert，其实它是弹出一个警告框。

2.对于各种弹出框，我们给出一个列表：

方法	描述
alert	弹出警告框
confirm	弹出确认框
prompt	弹出输入框

3.对于 alert()，我想朋友们是比较熟悉了，接下来我们来研究一下下面两个。confirm 是可以捕获到用户点击的是哪个按钮，也就是该函数会返回一个布尔值。而 prompt 是可以捕获到用户输入的文本信息的，也就是它会返回一个字符串。

4.我们在 star.html 中输入内容如下：

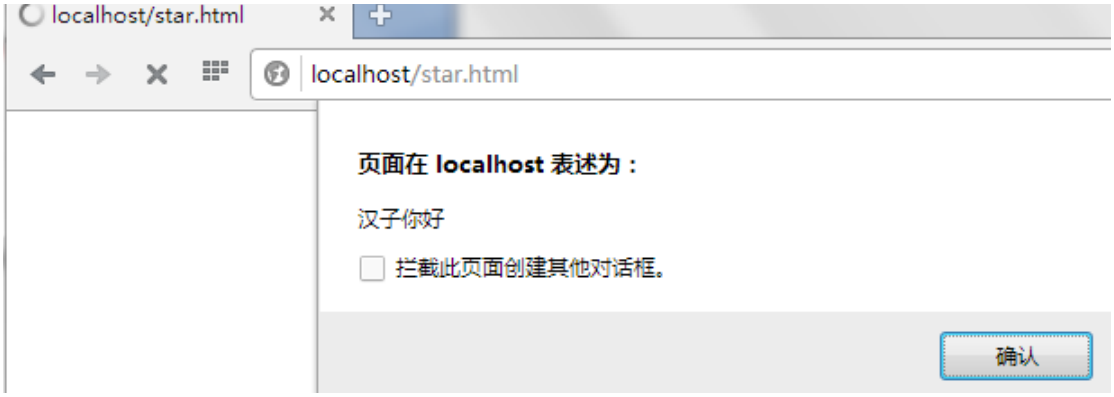
```
<html>
    <body>
        <script src="star.js"></script>
    </body>
</html>
```

5.然后我们在 star.js 中输入如下内容：

```
var msg = confirm("您是妹子吗？ ");
if(msg==true){
    alert("妹子你好");
}else{
    alert("汉子你好");
}
```

6.当我们进入页面的时候，看到效果如下：

7.当我们点击“取消”的时候，效果如下：

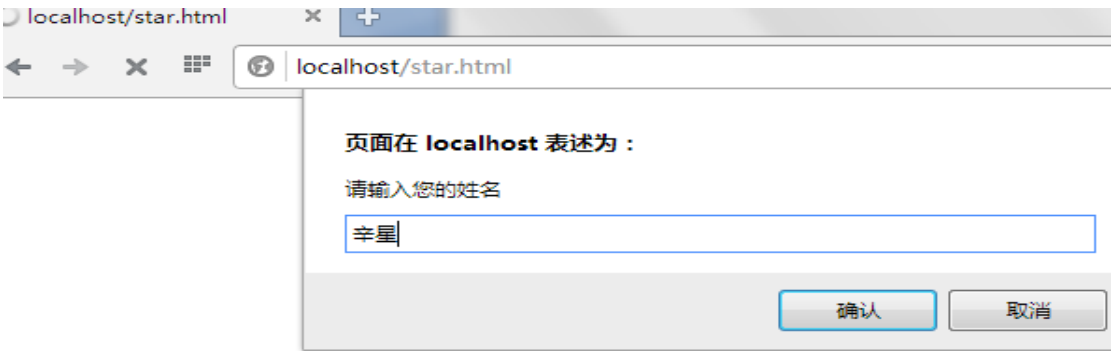


8.当然啦，如果我们点击确定，他就会出现“妹子你好”。

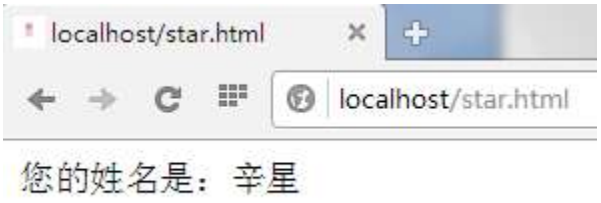
9.那么我们依然保持 star.html 内容不动，然后我们修改 star.js 内容如下：

```
var msg = prompt("请输入您的姓名");
document.write("您的姓名是: "+msg);
```

10.那么当我们进入该页面的时候，首先会看到，然后我们在输入框中输入自己的姓名，效果如下：



11.然后我们会看到如下效果：



12.近些年，说实话，弹出窗口很少见到了，为什么呢？原因就是因为弹出窗口实在是太烦了。

>>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<<<<<<

1.对于 BOM，虽然有一个明确的标准，但是由于各大浏览器厂商并且真正的协同工作，因此会有浏览器兼容上的一些问题。

2.由于本系列教程尚不成熟，可能有不少问题，因此，期待您的建议，让它更加完善。

3.辛星，期待您的关注。

第四部分：Ajax

第一节：简介 Ajax

>>>>>>>>>>>>概念<<<<<<<<<<<<<<<<<<<<<<<

1.首先 Ajax 是 Asynchronous Javascript and Xml 的简称，翻译成汉语就是异步的 Javascript 和 xml。

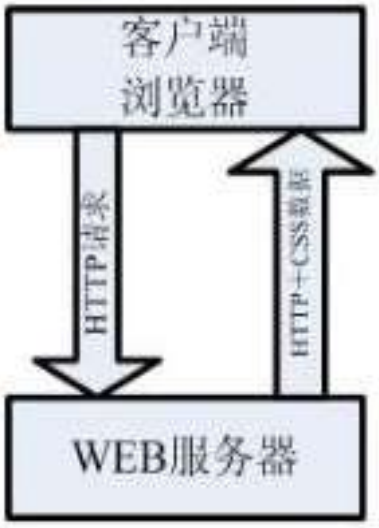
2.如果非要用汉语给出 Ajax 的音译的话，可以读作“阿贾克斯”。

3.然后 Ajax 是前端与服务器端交换数据并更新部分网页的一门技术，它的亮点就是在不重新加载整个网页的情况下可以刷新数据。

4.它不是一门新的编程语言，它是对现有技术的灵活运用。

>>>>>>>工作流程<<<<<<<<<<<<<<<<<<<<<<<

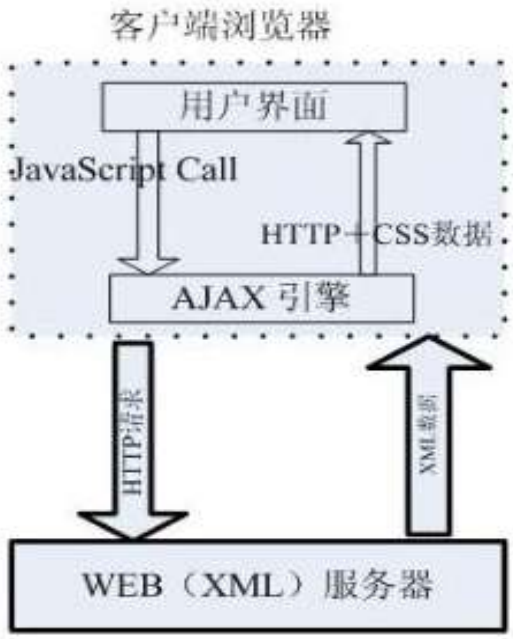
1.要弄清 ajax 的工作原理，我从网上找了几张图，首先是没有 ajax 的时候的简图：



2.对于这种工作流程，我们是再熟悉不过了，它就是前端和后端的交互中最简单的方式了。就是浏览器发送请求给服务器，然后服务器把数据发给浏览器，然后浏览器去解析这些数据。

3.可以看出这种工作方式，就是每次要更新一些数据，都必须重新发送一次请求，那样就会导致重新刷新一次页面。其实这也引出了 ajax 最引以为豪的一个特点“无刷新更新信息”。

4.然后就是当有 ajax 的时候，它的大致流程是这样的：



5.我们可以看出，它的工作大致是这样的：在 web 服务器和用户的浏览器之间加了一个 ajax 引擎，它是两者中间的纽带，当我们需要更新数据的时候，我们把数据给 ajax 引擎，它负责和后台交互，它的一个直接好处就是不用刷新用户界面。

>>>>>>>>>>>>具体实现<<<<<<<<<<<<<<<<<<<

1.那么 ajax 是怎么实现的呢？下面我们从代码层面来分析一下它的实现机制。

2.首先我们需要创建一个 XMLHttpRequest 对象，一般都喜欢简称为 xhr 对象，该对象用于和服务器端进行数据的交互。

3.然后 xhr 对象可以向服务器端发送请求，当然这个请求可以是携带数据过去的。

4.接下来就可以根据 xhr 的状态，来做出相应的判断，比如为 4 表示请求完成，响应就绪等等。

5.最后我们在现在网页上展示已有的数据就可以了。

>>>>>>>>>>>>小结<<<<<<<<<<<<<<<<<<<

1.本节课我们讲了一下 ajax 的大致原理。

2.对于它的实现机制和工作流程，我们可以在阅读完后续内容后再回过头来看，理解会更加深刻。

第二节：XMLHttpRequest 对象

>>>>>>>>>>>对象的创建<<<<<<<<<<<<<<<<<<<<<

1.所有的现代浏览器都支持 XMLHttpRequest 对象，它的创建语法如下：

```
var xhr  = new  XMLHttpRequest();
```

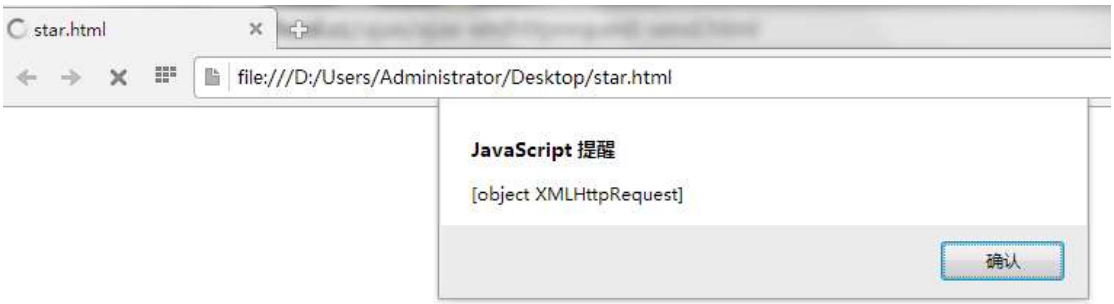
2.我们在 star.js 中书写如下代码：

```
var xhr = new XMLHttpRequest();
alert(xhr);
```

3.然后在 star.html 中书写如下代码：

```
<html>
    <body>
        <script src="star.js"></script>
    </body>
</html>
```

4.然后来看下效果吧：



5.这个对象是用来和服务器交换数据用的，也就意味着我们可以在不加载整个网页的情况下，对网页的某部分进行更新。

6.接下来我们就开始利用该对象来进行接受和发送数据了，此时又该怎么样呢?我们拭目以待吧。

>>>>>>请求服务器<<<<<<<<<<<<<<<<<

1.如果我们需要将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open()和 send()方法，这两个方法的作用是不同的。

2.我们先说 open 方法，它有三个参数：①第一个规定请求的类型，通常是 GET 或者 POST。②第二个是提交的 url 地址。③第三个是一个布尔类型变量，是否是异步处理请求，为 true 表示异步。

3.然后再说 send 方法，它用于当 open 方法中请求类型为 POST 的时候传递数据使用。

>>>>>>>设置状态<<<<<<<<<<<<<<<<<<<<<<<<

1.一般我们使用 open 方法的时候都设置为异步，也就是说，我们的数据什么时候到达是不一定的。

2.那么这个时候我们就要根据这个状态来做出相应的反应即刻，比如说数据到达了之后我们展示出来相应的数据。

3.我们通过给 XMLHttpRequest 对象的 onreadystatechange 属性设置一个方法，它通常是一个函数，通过判断属性来做出相应的反应。

4.对于状态，比较有代表性的一个就是 readyState，它会自动保存 XMLHttpRequest 的状态，它的值从 0 到 4，下面我们给出这 5 个值的一个列表：

取值	含义
0	请求尚未初始化
1	服务器连接已建立
2	请求已被后端接收
3	请求正在被处理
4	请求已完成，响应完毕

5.我们通常比较关注的就是状态为 4 的时候的情况，它通常意味着从后台拿到了相应的数据。

6.还有一个就是 status 这个，它的数据是由 HTTP 协议来规定的。

7.我们再回过头来看这个 onreadystatechange 的属性所具有的函数，它是一个回调函数，也就是说一旦状态发生变化，它会自动被调用。

>>>>>>>>>>返回数据<<<<<<<<<<<<<<<<<<<<

1.一般来说，如果服务器有了响应，那么这里通常是指有数据发送过来。

2.我们可以用 responseText 和 responseXML 来捕获这个数据，所不同的是，如果是 responseText，那么是以字符串形式的数据，如果是 responseXML，那么是 xml 形式的响应数据。

3.如果说服务器返回的并非 xml 格式的数据，我们使用 responseText 来接受就可以了。

>>>>>>>>>>ajax 实例演示<<<<<<<<<<<<<<<<<<<<

1.接下来我们看一个 ajax 的实例吧。不过我们这里必须要使用服务器端的一些知识，因为 ajax 并不是单纯的前台的工作，它是需要和后台交互完成的。

2.我们这里依然使用的是 php 语言，服务器我使用的是 Apache，不过影响不是很大，我会给大家解释 php 语言都干了什么。

3.我们首先在服务器文档根目录下创建一个 star.html 文件，书写如下代码：

```
<html>
    <body>
        <p>接下来我们用ajax的方式来计算1和2的和</p>
        <button onclick=cal()>计算</button>
        <p id="sum"></p>
        <script src="star.js"></script>
    </body>
</html>
```

4.这里我们先写了一个段落，然后我们做了一个按钮，它被点击的时候会调用 cal 方法，然后一个空的段落用来接收数据的。

5.我们在同级目录下新建一个 star.js 文件，书写如下代码：

```
var xhr = new XMLHttpRequest();
//当从服务器端拿到数据时，我们在sum那个段落中进行展示
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4 && xhr.status == 200){
        document.getElementById("sum").innerHTML = xhr.responseText;

    }
}
//它的功能就是发送数据
function cal(){
    xhr.open("GET","star.php?a=1&b=2",true);
    xhr.send();
}
```

6.可以看到它首先创建了一个 XMLHttpRequest 对象，然后当它的状态发生变化时，我们使用一个匿名函数来修改它的状态，这里当成功接收到数据之后，我们就把 id 为 sum 的这个段落的 html 文本内容修改为返回的数据。

7.然后就是 cal 函数，它的功能就是打开 star.php 文件，并且用 GET 的方式来发送两组数据，而且是异步的。

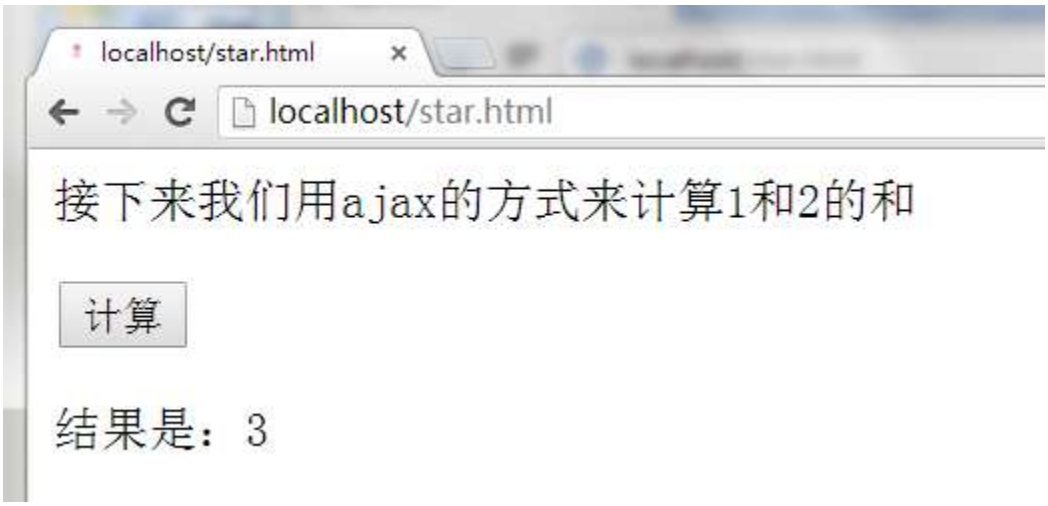
8.我们在同级目录下新建一个 star.php 文件，书写代码如下：

```
<?php
$sum = $_GET['a']+$_GET['b'];
echo "结果是: ",$sum;
```

9.这三行代码的意思就是说我们把以 GET 方式传递过来的 a 和 b 这两个属性的值进行相加，记为$sum，然后返回的数据就是“结果是：$sum”,这里的$sum 就是计算后的结果啦。

10.那么接下来我们看下效果图吧，首先我们在浏览器中输入 localhost/star.html，然后看到效果图如下：

11.然后我们点击“计算”这个按钮，会看到效果图如下：



12.这里读者可能看不到，它确实是无刷新的给出了结果，然后确实也和我们想象的是一样的。

>>>>>>>>>>浏览器兼容性<<<<<<<<<<<<<<<<<<<<

1.这个时候不得不说浏览器的兼容性问题了，因为这个问题对于咱们前端工作人员来说，很是个问题。

2.就是这个 ajax 这里，我们就会看到经常书上会写，如果是 IE5 和 IE6，我们应该怎么做，其实这里我感觉不是那么重要了，因为都 2014 年了，很少会有 IE5 和 IE6 的出现了。

3.记得有位大师做的很在理“如果你的公司坚持要兼容特别陈旧的产品，你应该考虑是否要换家公司”。

4.其实我也这么想：一家公司那么守旧，它的创新能力究竟怎样？

>>>>>>>>>>总结<<<<<<<<<<<<<<<<<<<<<<<<

1.本节主要是给出了 ajax 的整体的实现流程，它的实现首先从 XMLHttpRequest 对象创建开始。

2.然后我们需要设置它的状态发生变化的时候应该怎么做，这里重点关注一下 readySate 和 status 这两个值。

3.最后就是使用 open 和 send 这两个方法来发送数据，这点希望朋友们可以注意下。

4.之后就是使用 responseText 或者 responseXML 来接收数据。

5.实例的话，我上面给了一个，更多的例子，朋友们可以自己去研究下奥。

6.其实这些东西都是不学不会，一学就会的，毕竟，它的技术含量并不算很高。

我们的疑惑，求解答

>>>>>>联系方式<<<<<<<<<<<<<<<<<<<<<<<<

1.其实在开始部分已经介绍的比较清楚了，我推荐发邮件给 xinguimeng@163.com，当然也可以加我 QQ：1808347923，来，我们面对面谈谈吧。

2.如果您想加入我们的大家庭，也可以加入 QQ 群：392328871 奥，我们一起来交流吧。

>>>>>>>>>>我们的困惑<<<<<<<<<<<<<<<<<<<<

1.如果您感觉本书有什么问题，哪里讲的不清楚，或者您看到哪里感觉看不下去之类的，都可以给我们联系。

2.当然，由于 Javascript 是一个比较大的话题，我们可能在后续书目中拆成几本来讲，您的意见是什么呢？

3.期待您的参与，您的支持是我们前进的动力。

>>>>>>>>>>>>关于我<<<<<<<<<<<<<<<<<<

1.也有些人对我有点好奇，我只能说：我只是，一个帅哥。

2.一般来说，你可以把我当做大哥哥，叫我“星哥”就可以啦。