### 第一个JS文件

使得.html文件与.js文件分离，一般一个html配一个js文件

html文件

|  |
| --- |
| <html>  <head>  <title>JavaScript程序开发</title>  <meta charset="utf-8">  <script type="text/javascript"  src="js/testjs.js"> //src表示对url的引用，这里连接到js文件夹下的test.js文档  //在引用.js文档后，本来在script中的语句会不被读取  //window.alert("hellow world!");  //document.write("nihao");  </script>  </head>  <body>    </body>  </html> |

javascript文件

|  |
| --- |
| temp = "hello";  for (x = 0 ; x < 100 ; x ++){  temp += x;  }  document.write("<h1>");  document.write(temp);  document.write("</h1>"); |

### 认识JavaScript的事件处理

Html

|  |
| --- |
| <html>  <head>  <title>JavaScript程序开发</title>  <meta charset="utf-8">  <script type="text/javascript"  src="js/testjs.js"> //src表示对url的引用，这里连接到js文件夹下的test.js文档  //在引用.js文档后，本来在script中的语句会不被读取  //window.alert("hellow world!");  //document.write("nihao");  </script>  </head>  <body onload="loadFun()" onunload="unloadFun()"> //onload和onunload表示打开和关闭页面    </body>  </html> |

JavaScript

|  |
| --- |
| function loadFun(){  alert("欢迎！") ;  }  function unloadFun(){  alert("再见！") ;  } |

注意，所有的JavaScript事件都会以“on”开头。如果一个操作可能触发n个函数，则函数之间加上“；”。

|  |
| --- |
| <body onload="loadFun();unloadFun()" onunload="unloadFun()"> |

### 动态事件绑定

从开发角度来说，所有的JavaScript代码都应储存在\*.js文件中。那么在html文件中定义事件就会出现问题，那么就需要采用动态绑定的方式，在页面加载的时候绑定。

范例：编写页面绑定操作

Html

|  |
| --- |
| <html>  <head>  <title>JavaScript程序开发</title>  <meta charset="utf-8">  <script type="text/javascript"  src="js/testjs.js">  </script>  </head>  <body>    </body>  </html> |

JavaScript

|  |
| --- |
| window.onload = function() { //为onload事件绑定处理  loadFun() ; //调用页面加载的处理函数，前端页面没有js内容  }  function loadFun() {  alert("页面加载了！") ;  } |

以上是以页面载入为事件触发条件的编程，但在实际中，事件有很多种，因此要有多种绑定事件的方法。

如果要为一个元素绑定事件，那么必须保证这个元素有一个绝对的独一无二的ID属性。

对于html中的显示元素，要对其添加事件则一定需要一个id。

动态页面绑定操作实例

Html

|  |
| --- |
| <html>  <head>  <title>JavaScript程序开发</title>  <meta charset="utf-8">  <script type="text/javascript"  src="js/testjs.js">  //在引用.js文档后，本来在script中的语句会不被读取  //window.alert("hellow world!");  //document.write("nihao");  </script>  </head>  <body>  <button id="mybut"><h1>PUSH ME</h1></button>  </body>  </html> |

JavaScript

|  |
| --- |
| function fun() {  alert("按钮被点击了！") ;  }  window.onload = function(){ //为onload事件绑定处理  //在JavaScript里所有的元素都属于对象，js是基于对象的编程，即只能操作html中已有的对象，不可自己新定义对象  //如果要取得对象，可以使用document.getElementById("ID名称")  var obj = document.getElementById("mybut") ;  if (obj != undefined) { //表示此对项已明确存在了  // 对象.addEventListener(事件类型,处理函数,false)  //动态绑定事件时不要加“on”  obj.addEventListener("click",function(){  fun() ;  },false) ;  }  } |

以上的JavaScript动态事件绑定为最普遍的处理模式。

### 鼠标事件

Html

|  |
| --- |
| <img src="images/pic-1.jpg" id="mypic" style="width: 800"> |

JavaScript

|  |
| --- |
| window.onload = function(){ //为onload事件绑定处理  document.getElementById('mypic').addEventListener('mousedown',function(){  alert('鼠标事件触发') ;  },false) ;  } |

### 图品展示效果

用按钮控制图片的上一张，下一张功能，使用click单击事件。

所有的图片放在images文件夹中，并采取“pic-\*.jpg”形式来命名，“\*”取值范围为1-6。

|  |
| --- |
| <body>  <div id = "showPic">  <img src="images/pic-1.jpg" id="mypic" style="height: 500">  </div>  <div id = "ctrlPic">  <button id = previousBut><h1>上一张</h1></button>  <button id = nextBut><h1>下一张</h1></button>  </div>  </body> |

JavaScript

|  |
| --- |
| var curr = 1 ; //默认从第一张开始显示  function changePic(){ //切换图片显示  var imgObj = document.getElementById('mypic') ;  imgObj.src = 'images/pic-' + curr + '.jpg' ; //操作src属性  }  window.onload = function(){ //为onload事件绑定处理  document.getElementById('previousBut').addEventListener('click',function(){  if (curr > 1) {  curr -- ;  }else{  curr = 6 ;  }  changePic() ;  },false) ;  document.getElementById('nextBut').addEventListener('click',function(){  if (curr < 6) {  curr ++ ;  }else{  curr = 1 ;  }  changePic() ;  },false) ;  } |

也可设置为图片自动切换，使用setTimeout（执行函数，毫秒单位间隔）函数

|  |
| --- |
| <body>  <div id = "showPic">  <img src="images/pic-1.jpg" id="mypic" style="height: 500">  </div>    </body> |

JavaScript

|  |
| --- |
| var curr = 1 ; //默认从第一张开始显示  function changePic(){ //切换图片显示  if (curr < 6) {  curr ++ ;  } else{  curr = 1 ; //已经显示完了，从头开始  }  var imgObj = document.getElementById('mypic') ;  imgObj.src = 'images/pic-' + curr + '.jpg' ; //操作src属性  setTimeout('changePic()',2000) ;  }  window.onload = function(){ //为onload事件绑定处理  changePic() ; //两秒后调用changePic()函数  } |

**总结：**每一个HTML元素在JavaScript中都表示一个对象，对象可以调用里面的属性。

每一个元素都可以用addEventLisenter（）函数动态绑定，且都是在页面加载之后绑

定。

取得元素对象：document.getElementById（‘ID’）名称。

### HTML元素取得（表单操作）

每一个HTML元素都可以用id进行标注，之后用document.getElementById（）来获取元素对象在进行操作。

但是document.getElementById（）只能够取得一个元素信息。

|  |
| --- |
| <body>  <div id = "infoTab">  <table border='1' cellpadding="5" cellspacing="0" bgcolor="F2F2F2" width="100%">  <tr id='infoTr'>  <td>姓名</td>  <td>年龄</td>  <td>学历</td>  </tr>  <tr id='infoTr'>  <td>张1</td>  <td>1</td>  <td>小学</td>  </tr>  <tr id='infoTr'>  <td>张2</td>  <td>2</td>  <td>小学</td>  </tr>  <tr id='infoTr'>  <td>张3</td>  <td>3</td>  <td>小学</td>  </tr>  </table>  </div>  </body> |

在上代码构建的表格中，想实现鼠标经过改变颜色的功能，最方便的方法为--由于所有行使用一个id标注“id = infoTr“。

|  |
| --- |
| /\*\*  \* obj:表示当前要改变颜色的对象  \* color：设置要显示的颜色种类  \*/  function changeColor(obj,color){  obj.bgColor = color ;  }  window.onload = function(){  var trObj = document.getElementById('infoTr') ;  trObj.addEventListenter('mouseover',function(){  //this表示当前操作的对象，即tr  changeColor(this,'FFFFFF') ;  },false) ;  trObj.addEventListenter('mouseout',function(){  changeColor(this,'F2F2F2') ;  },false) ;  } |

运行后发现代码只对表格第一行起效，问题在于“infoTr“的id包含多个元素，及多行表格，但document.addEventListener（）函数只能取第一个元素，要想取得多个，应采用document.all（’infoTr’）的形式，即当id有重复时，返回的时对象数组。

|  |
| --- |
| function changeColor(obj,color){  obj.bgColor = color ;  }  window.onload = function(){  var trObj = document.all('infoTr') ;  for (var x = 0; x < trObj.length ; x++) {  trObj[x].addEventListener('mouseover',function(){  //this表示当前操作的对象，即tr  changeColor(this,'FFFFFF') ;  },false) ;  trObj[x].addEventListener('mouseout',function(){  changeColor(this,'F2F2F2') ;  },false) ;  }  } |

现阶段要掌握的是，使用document.getElementById（）只返回元素的**第一个**对象，使用document.all（）则以**数组形式**返回元素的所有对象。

### 基础验证

在用户填写表单的过程中需要提交数据，且数据要符合输入要求。

如果要进行验证，先对输入的数据进行验证处理。

定义基础表单

Form.html文件

|  |
| --- |
| <body>  <form action="pass.html" method="post" id="loginForm">  <!-- 在以后的开发过程中，要保证name与id属性内容完全一致，否则会出现怪异问题-->  登录邮箱：<input type="text" name="email" id="email"><br>  <button type="submit" id="subBut"><h1>登录</h1></button>  </form>  </body> |

Pass.html文件

|  |
| --- |
| <body>  <h1>表单验证通过！！！</h1>  </body> |

随后要对表单数据进行验证，验证须在form.js中完成，习惯做法是找到登录按钮进行验证。

|  |
| --- |
| window.onload = function(){ //在页面加载完成后进行动态事件绑定  document.getElementById('subBut').addEventListener('click',function(){  var emailObj = document.getElementById('email') ;  if (emailObj.value == '') {  alert('您还没有输入登录邮箱，无法登录！')  } else{ //输入正确，理论上可以提交  }  },false) ;  } |

以上代码为最简单的验证处理，但无法拦截表单是否通过，要完成拦截功能则需要onsubmit函数，若返回true，则提交表单，若返回false，则不提交表单。

|  |  |
| --- | --- |
| function f(){  return false ;  } | <form action="pass.html" method="post" id="loginForm" onsubmit="return f()"> |

此时“<form>“元素中的”onsubmit="return f()"“表示接受f（）的返回结果。若返回true，则提交表单，若返回false，则不提交表单。

但这种操作属于静态的绑定事件处理，所以无法使用。只能通过动态事件绑定处理，在表单上设置submit事件，再利用“表单对象.submit（）”函数手动提交表单。

|  |
| --- |
| window.onload = function(){ //在页面加载完成后进行动态事件绑定  document.getElementById('loginForm').addEventListener('submit',function(){  var emailObj = document.getElementById('email') ;  if (emailObj.value == '') {  alert('您还没有输入登录邮箱，无法登录！')  } else{ //输入正确，理论上可以提交  this.submit() ; //当前元素提交表单  }  },false) ;  } |

上代码可以实现动态绑定，但不标准，因为要求输入邮箱，不能是其他格式。经过动态绑定后的代码如下，通过与拦截都由js文件实现。

Html

|  |
| --- |
| <body>  <form action="pass.html" method="post" id="loginForm" >  <!-- 在以后的开发过程中，要保证name与id属性内容完全一致，否则会出现怪异问题-->  登录邮箱：<input type="text" name="email" id="email"><br>  <button type="submit" id="subBut"><h1>登录</h1></button>  </form>  </body> |

JavaScript

|  |
| --- |
| window.onload = function(){ //在页面加载完成后进行动态事件绑定  document.getElementById('loginForm').addEventListener('submit',function(e){  console.log(e) ;  var emailObj = document.getElementById('email') ;  if (emailObj.value == '') {  alert('您还没有输入登录邮箱，无法登录！') ;  **if (e && e.preventDefault) { //现在是w3c标准下执行**  **e.preventDefault() ; //阻止浏览器的动作**  **} else { //专门针对ie浏览器的处理**  **window.event.returnValue = false ;**  **}**  } else{  if (/^\w+@\w+\.\w+$/.test(emailObj.value)) { //输入正确，理论上可以提交  this.submit() ; //当前元素提交表单  } else{  alert('请输入合法的EMAIL地址！') ;  if (e && e.preventDefault) { //现在是w3c标准下执行  e.preventDefault() ; //阻止浏览器的动作  } else { //专门针对ie浏览器的处理  window.event.returnValue = false ;  }  }  }  },false) ;  } |

以上代码能完成目的，但不实用，以下为

### 合理验证规划

要更好地进行数据操作，需要模块化设计。

设计util.js文件负责校对输入格式。

|  |
| --- |
|  |

建立一个css/form.css文件

|  |
| --- |
| .success {  background: #f5f5f5 ;  font-weight: bold ;  color: #000000 ;  border: 1px #009900 solid ;  }  .failure {  background: #f5f5f5 ;  font-weight: bold ;  color: #000000 ;  border: 1px #009900 solid ;  }  .init {  background: #f5f5f5 ;  font-weight: bold ;  color: #000000 ;  } |

对于验证而言，往往是在一个组件失去焦点后。失去焦点为blur事件，获得焦点为focus。

以下是第13讲到25分钟为止的内容，后面内容为，增加密码确认功能，id增加正则判断功能等

所用文件为vform.html;vform.js;util.js;form.css;pass.html

### 单选按钮事件处理

单选钮属于多选一处理流程，但是由于单选钮也是HTML元素，所以对Javascript而言也表示对象。

|  |
| --- |
| <input type="radio" name="sex" id="**sex**" value="male - 男">男  <input type="radio" name="sex" id="**sex**" value="female - 女">女 |

单选钮的特点是名字一定要相同才可以实现，所以此处如果名字相同，那么就是数组

Document.all（）取得。单选钮有一种默认选中的状态，ckecked属性，返回的是boolean型。

常见的单选钮处理操作如下

|  |
| --- |
| <html>  <head>  <title>JavaScript程序开发</title>  <meta charset="utf-8">  <script type="text/javascript">  window.onload = function () {  document.getElementById('showBut').addEventListener("click",function(){  var sexObj = document.all("sex") ;  if (sexObj[0].checked) {  alert("性别是：" + sexObj[0].value) ;  } else {  alert("性别是：" + sexObj[1].value) ;  }    },false) ;  }  </script>  </head>  <body>  <form action="">  性别： <input type="radio" name="sex" id="sex" value="male - 男" checked>男  <input type="radio" name="sex" id="sex" value="female - 女">女<br>  <input type="button" value="选择" id="showBut">  </form>  </body>  </html> |

### 复选框处理

复选框本身也是多个组件名字相同。所以在定义复选框时依然要来考虑使用document.all()取得全部内容。

|  |
| --- |
| <html>  <head>  <title>JavaScript程序开发</title>  <meta charset="utf-8">  <script type="text/javascript">  window.onload = function () {  document.getElementById('showBut').addEventListener("click",function(){  var instObj = document.all("inst") ;  var value = "兴趣为：" ;  for (var x = 0 ; x < instObj.length ; x ++)  {  if (instObj[x].checked) {  value += instObj[x].value+"、" ;  }  }  alert(value) ;  },false) ;  }  </script>  </head>  <body>  <form action="">  王维暄的爱好：<br>  <input type="checkBox" id="inst" name="inst" value="犯贱">犯贱<br>  <input type="checkBox" id="inst" name="inst" value="家春">家春<br>  <input type="checkBox" id="inst" name="inst" value="抢小孩棒棒糖">抢小孩棒棒糖<br>  <input type="checkBox" id="inst" name="inst" value="偷看三岁孩子洗澡">偷看三岁孩子洗澡<br>  <input type="checkBox" id="inst" name="inst" value="猥琐男">猥琐男<br>  <input type="checkBox" id="inst" name="inst" value="龌龊宅">龌龊宅<br>  <input type="checkBox" id="inst" name="inst" value="不是好人">不是好人<br>  <input type="button" value="选择" id="showBut">  </form>  </body>  </html> |

但可能出现全选情况，因此要使得每一个复选框的checked状态跟随当前的selCheck的状态。

|  |
| --- |
| document.getElementById("selCheck").addEventListener("click",function(){  var instObj = document.all("inst") ;  if (instObj.length == undefined){ //只有一个元素  instObj.checked = this.checked ;  } else {  for (var x = 0 ; x < instObj.length ; x ++)  {  //每一个复选框的checked状态跟随当前的selCheck的状态  instObj[x].checked = this.checked ;  }  } |

还要注意的一点是，document.all取得的是数组形式，如果对象只有一个元素，虽可以取回元素，但无法组成数组，此时“元素.length”为”undefined”状态。

### 下拉列表框处理

下拉列表框处理操作主要使用onchange的事件，此事件描述的是内容改变后的行为。

|  |
| --- |
| <html>  <head>  <title>JavaScript程序开发</title>  <meta charset="utf-8">  <script type="text/javascript">  window.onload = function () {  document.getElementById("city").addEventListener("**change**",function(){  alert("当前选项为：" + this.value) ;  },false) ;  }  </script>  </head>  <body>  <form action="">  你来自哪里：  <select id="city">  <option value="BJ-北京">北京</option>  <option value="NJ-南京">南京</option>  <option value="TJ-天津">天津</option>  <option value="WH-武汉">武汉</option>  <option value="WH-威海">威海</option>  </select>  <!--<input type="button" value="选择" id="showBut">-->  </form>  </body>  </html> |

以上为最基础下拉列表代码。

### 文本域事件处理

文本域可以输入大量文字信息，文本域上有一些键盘可以处理的事件 keydown，keypress，keyup等

|  |
| --- |
| <html>  <head>  <title>JavaScript教程</title>  <meta charset="utf-8">  <script type="text/javascript">  window.onload = function(){  document.getElementById("msg").addEventListener("keypress",function(){  document.getElementById("noteSpan").innerHTML += "keypress<br>" ;  },false) ;  document.getElementById("msg").addEventListener("keydown",function(){  document.getElementById("noteSpan").innerHTML += "keydown<br>" ;  },false) ;  document.getElementById("msg").addEventListener("keyup",function(){  document.getElementById("noteSpan").innerHTML += "keyup<br>" ;  },false) ;  }  </script>  </head>  <body>  <form action="">  请输入你对王的评价：<br>  <textarea id="msg" cols="30" rows="10"></textarea>  <div><span id="noteSpan"></span></div>  </form>  </body>  </html> |

利用此功能可实现如，要求输入评论不超过20个字，如果是20个不允许提交表单。

|  |
| --- |
| <html>  <head>  <title>JavaScript教程</title>  <meta charset="utf-8">  <script type="text/javascript">  window.onload = function(){  document.getElementById("msg").addEventListener("keypress",function(){  cal() ;  },false) ;  document.getElementById("msg").addEventListener("keydown",function(){  cal() ;    },false) ;  document.getElementById("msg").addEventListener("keyup",function(){  cal() ;    },false) ;  }  function cal() { //计算输入文字长度  var obj = document.getElementById("msg") ;  var span = document.getElementById("noteSpan") ;  var max = 20 ; //最多文本字数  var cur = obj.value.length //当前的输入长度  if (cur <= max) {  span.innerHTML = "还允许输入"+(max-cur)+"个字" ;  document.getElementById("subBut").disabled=false ;  } else {  span.innerHTML = "输入文字过多，超过了<font color='red'>"+(max-cur)+"</font>个字" ;  document.getElementById("subBut").disabled=true ;  }  }  </script>  </head>  <body>  <form action="">  <div>请输入你对王的评价：<br>  <textarea id="msg" cols="30" rows="3"></textarea></div>  <div><span id="noteSpan">还允许输入20个字</span></div>  <div>  <input type="submit" id="subBut" value="提交">  </div>  </form>  </body>  </html> |

以上为常见的设计形式，JavaScript还有一堆功能留给设计者使用，但对于开发者来说，掌握以上表单的处理形式是最重要的。

### Window对象使用

在JavaScript中window表示一个窗体对象，所以在整个事件处理中都是以弹框为主。

下例为弹窗警告

|  |
| --- |
| <script type="text/javascript">  window.alert("Hello World!") ;  </script> |

下例为数据输入操作

|  |
| --- |
| <script type="text/javascript">  var name = window.prompt("请输入姓名","你的姓名") ;  document.write("<h1>"+name+"</h1>") ;  </script> |

下例为在window对象里还可以打开其他子窗口。

|  |
| --- |
| <meta charset="utf-8">  <script type="text/javascript">  window.open("pass.html") ;  </script> |

下例为window窗口中的确认框----删除确认

|  |
| --- |
| <meta charset="utf-8">  <script type="text/javascript">  window.onload = function(){  document.getElementById("delBut").addEventListener("click",function(){  if (window.confirm("确认要删除吗？")) {  window.location = "pass.html" ; //跳转到指定路径  }  },false) ;  }  </script>  <button id="delBut">删除邮件</button> |

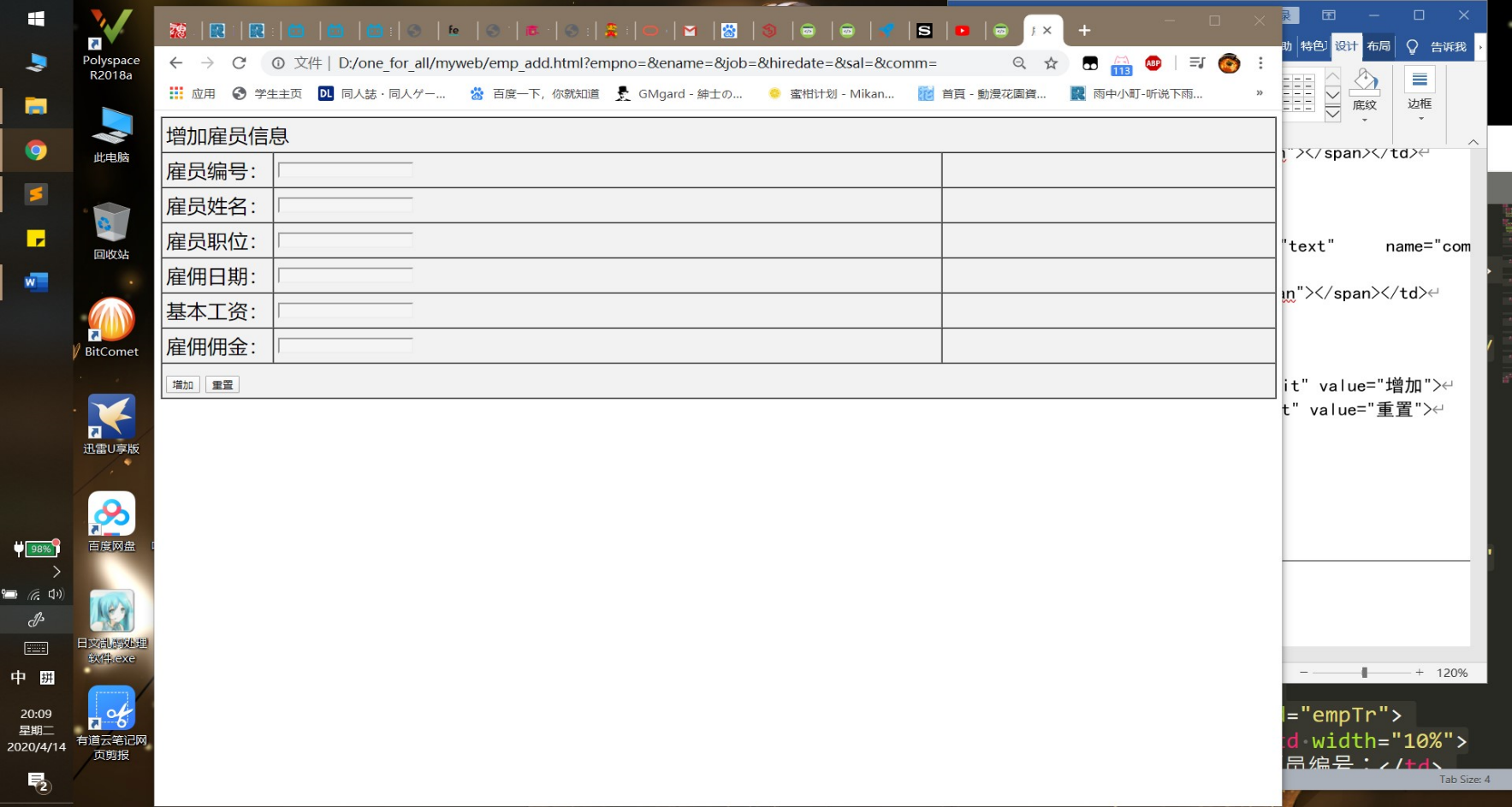
### 定义表单验证

按要求定义一个雇员信息增加页面，要求为：

* 雇员编号：必须是4位，按照正则进行校验
* 雇员姓名：不能为空
* 雇员职位：不能为空
* 雇佣日期：按“yyyy-mm-dd”格式编写，按照正则进行校验
* 基本工资：按照小数编写，小数最多2位(整数最多5位)，按照正则进行校验
* 佣金：按照小数编写，小数最多2位(整数最多5位)，按照正则进行校验

定义表单：将form.css导入cssmulu；在emp\_add.html中编写表单，以及导入form.css文件引用。

|  |
| --- |
| <html>  <head>  <title>雇员管理程序</title>  <meta charset="utf-8">  <link rel="stylesheet" type="text/css" href="css/form.css">  </head>  <body>  <form id="emoForm">  <table width="100%" cellpadding="5" cellspacing="0" border="1" bgcolor="#F2F2F2">  <tr>  <td colspan="3">增加雇员信息</td>  </tr>  <tr id="empTr">  <td width="10%">雇员编号：</td>  <td width="60%"><input type="text" name="empno" id="empno" class="init"></td>  <td width="30%"><span id="empnoSpan"></span></td>  </tr>  <tr id="empTr">  <td>雇员姓名：</td>  <td><input type="text" name="ename" id="ename" class="init"></td>  <td><span id="enameSpan"></span></td>  </tr>  <tr id="empTr">  <td>雇员职位：</td>  <td><input type="text" name="job" id="job" class="init"></td>  <td><span id="jobSpan"></span></td>  </tr>  <tr id="empTr">  <td>雇佣日期：</td>  <td><input type="text" name="hiredate" id="hiredate" class="init"></td>  <td><span id="hiredateSpan"></span></td>  </tr>  <tr id="empTr">  <td>基本工资：</td>  <td><input type="text" name="sal" id="sal" class="init"></td>  <td><span id="salSpan"></span></td>  </tr>  <tr id="empTr">  <td>雇佣佣金：</td>  <td><input type="text" name="comm" id="comm" class="init"></td>  <td><span id="commSpan"></span></td>  </tr>  <tr id="empTr">  <td colspan="3">  <input type="submit" value="增加">  <input type="reset" value="重置">  </td>  </tr>  </table>  </form>  </body>  </html> |



1.页面动态效果：为表格增加一些现显示的过渡效果。

* 建立js/util.js文件，其中要有表格改变的处理函数。
* 既然是在emp\_add.html文件进行处理，所以应该准备一个emp\_add.js文件。
* 动态设置显示的效果，增加mouseover、mouseout事件，在表格行元素中增加。

2.考虑到代码可重用性问题，将具体功能交给util.js完成。

* 建立validateEmpty（）、validateRegex（）、validateNumber（）、validateDate（）。
* 建立玩一个函数后一定要对此函数可用性进行测试。
* 在emp\_add.js中动态绑定事件，使用blur事件。

3.分别绑定完事件后，下面要针对于表单进行事件的处理。

### 房贷计算器

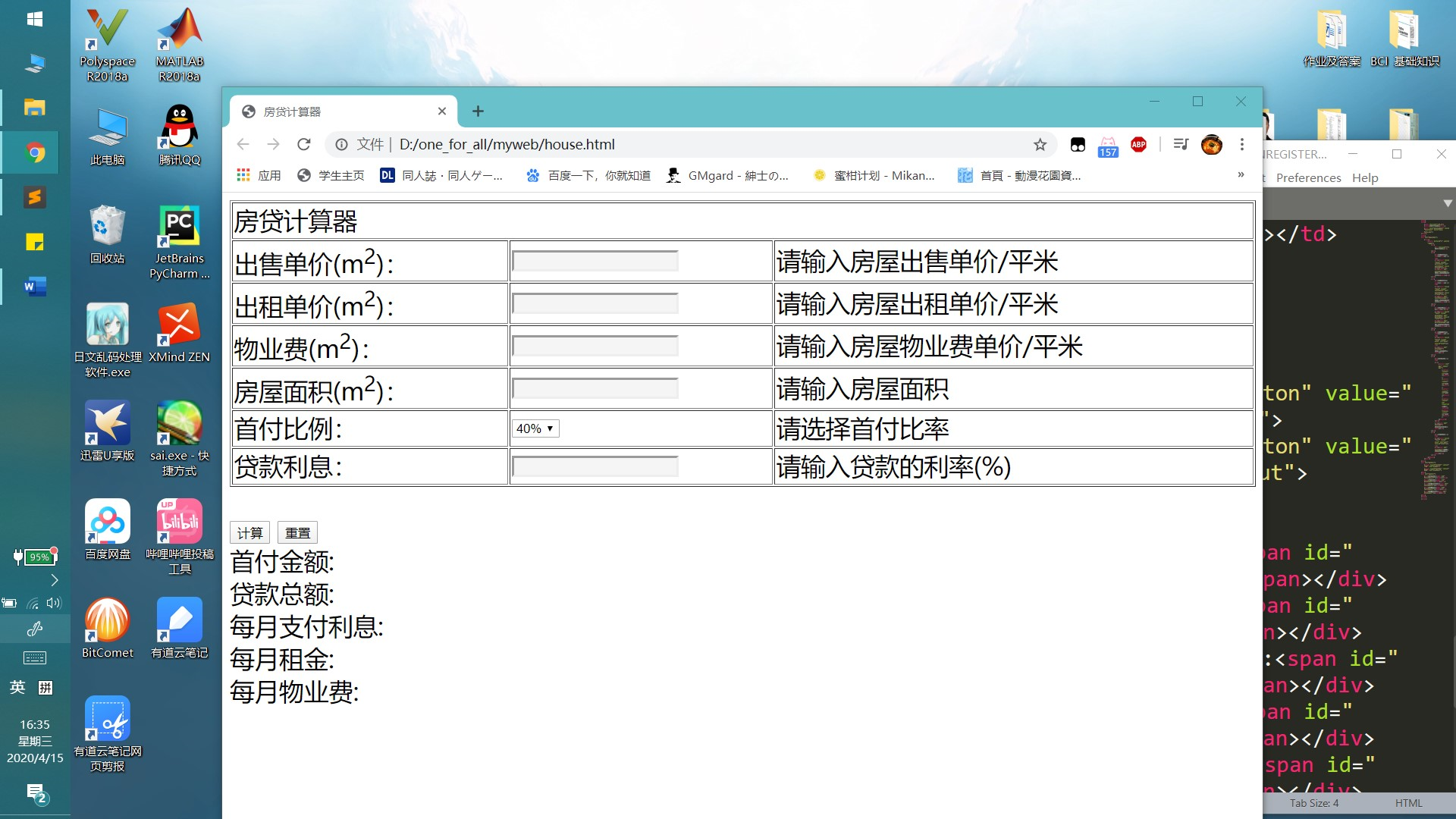
结合表单验证、数据计算。

1，实现表单界面分为三层：

* 第一层（inputDiv）;作为输入数据使用
* 第二层（butDiv）：控制按钮
* 第三层（擦了Div）：显示所有计算结果

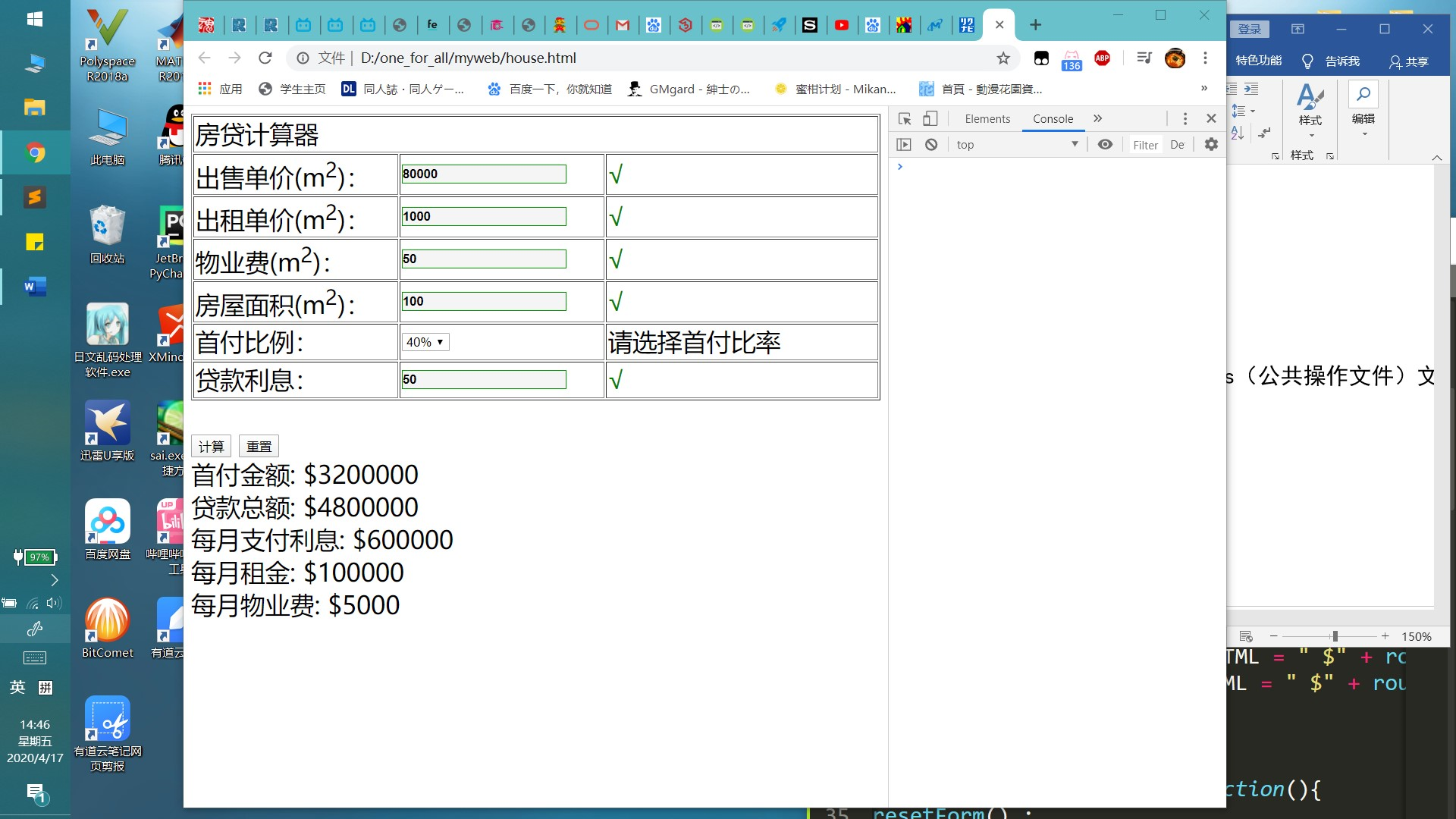
|  |
| --- |
| <html>  <head>  <meta charset="utf-8">  <title>房贷计算器</title>  <link rel="stylesheet" type="text/css" href="css/form.css">  </head>  <body>  <div id="inputDiv"></div>  <div id="butDiv"></div>  <div id="calDiv"></div>  </body>  </html> |

2，在代码中进行表单控件的填充操作



数据验证

建立house.js文件，建立house\_util.js（公共操作文件）文件。



详情参看 house.html,house.js,house\_util.js。