# 第3天 Javascript(上)

【学习目标】

## 1 掌握Javascript基本语法

a)掌握Javascript基本语法格式

b)掌握变量的定义和使用

c)掌握JS中的数据类型

d)掌握JS中语句的控制

## 2掌握Javascript对象获取方式

## 3掌握Javascript标签的基本操作

## 4会使用Javascript操作CSS样式

## 5使用Javascript编写各种事件

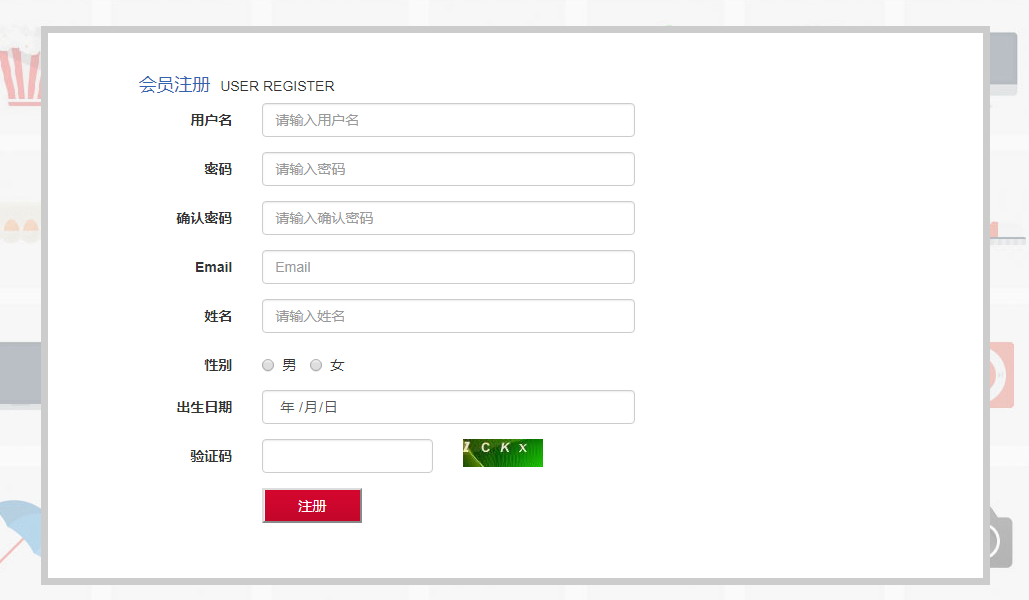
## 6使用Javascript编写定时程序

# 案例一:使用JS完成注册页面的校验

## 案例介绍

用户在提交表单是，需要对用户填写的数据进行校验。因为用户如果输入非法内容，则会导致服务器的压力过大，因此，一般提供前端校验和后台校验，前端校验防君子不防小人。

注册表单如下：

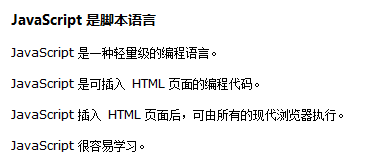


## 相关知识点

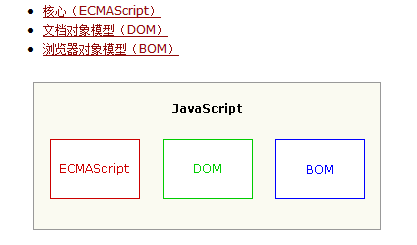
* + 1. JavaScript概述

**1.简介**

以下是W3School对JavaScript的简述:



JavaScript的组成:



**2.JavaScript特点:**

JavaScript脚本语言具有以下特点:

(1)[脚本语言](https://baike.baidu.com/item/%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80)。JavaScript是一种解释型的脚本语言,C、[C++](https://baike.baidu.com/item/C%2B%2B)等语言先[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91)后执行,而JavaScript是在程序的运行过程中逐行进行解释。

(2)基于对象。JavaScript是一种基于对象的脚本语言,它不仅可以创建对象，也能使用现有的对象。

(3)简单。JavaScript语言中采用的是弱类型的变量类型,对使用的数据类型未做出严格的要求,是基于Java基本语句和控制的脚本语言,其设计简单紧凑。

(4)动态性。JavaScript是一种采用事件驱动的脚本语言,它不需要经过Web服务器就可以对用户的输入做出响应。在访问一个网页时，鼠标在网页中进行鼠标点击或上下移、窗口移动等操作JavaScript都可直接对这些事件给出相应的响应。

(5)跨平台性。JavaScript脚本语言不依赖于操作系统,仅需要浏览器的支持。因此一个JavaScript脚本在编写后可以带到任意机器上使用,前提上机器上的浏览器支 持JavaScript脚本语言,目前JavaScript已被大多数的浏览器所支持。

不同于服务器端脚本语言，例如[PHP](https://baike.baidu.com/item/PHP/9337)与[ASP](https://baike.baidu.com/item/ASP/128906)，JavaScript主要被作为客户端脚本语言在用户的浏览器上运行，不需要服务器的支持。所以在早期程序员比较青睐于JavaScript以减少对服务器的负担，而与此同时也带来另一个问题：安全性。

而随着服务器的强壮，虽然程序员更喜欢运行于服务端的脚本以保证安全，但JavaScript仍然以其跨平台、容易上手等优势大行其道。同时，有些特殊功能（如[AJAX](https://baike.baidu.com/item/AJAX/8425)）必须依赖Javascript在客户端进行支持。随着引擎如V8和框架如[Node.js](https://baike.baidu.com/item/Node.js)的发展，及其事件驱动及[异步IO](https://baike.baidu.com/item/%E5%BC%82%E6%AD%A5IO)等特性，JavaScript逐渐被用来编写服务器端程序。

**3.JavaScript的作用**

使用JavaScript添加 页面动画效果，提供用户操作体验。主要应用有：嵌入动态文本于HTML页面，对浏览器事件作出响应，读写HTML元素，验证提交数据，检测访客的浏览器信息等。

**4.JavaScript的引入方式**

在HTML文件中引入JS有两种方式:

第一种:内嵌式.将JS代码直接嵌入HTML页面中。则该JS代码只对本页面有效。

使用方法：在HTML页面中使用<script></script>标签引入。如下：

|  |
| --- |
| <script>  //此处为JS代码  </script> |

第二种：外联式。独立称为一个以.js为后缀的文件，并在HTML页面中引入。

|  |
| --- |
| <script src="../js/main.js" type="text/javascript" charset="UTF-8"></script> |

* + 1. JS基本语法

**1.变量**

1）变量命名

与代数一样，JavaScript 变量可用于存放值（比如 x=2）和表达式（比如 z=x+y）。

变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age, sum, totalvolume）。

* 变量必须以字母开头
* 变量也能以 $ 和 \_ 符号开头（不过我们不推荐这么做）
* 变量名称对大小写敏感（y 和 Y 是不同的变量）

提示：JavaScript 语句和 JavaScript 变量都对大小写敏感。

2）变量的声明

var 变量名;//JS变量可以不声明，直接使用。默认值:undefined

3）变量的取值

var 变量名=值;//JS变量是弱类型，即同一个变量可以洗存放不同类型数据。

**2.数据类型**

【基本类型】

* undefined：Undefined 类型只有一个值，即 undefined。当声明的变量未初始化时，该变量的默认值是 undefined。
* Null 类型：另一种只有一个值的类型是 Null，它只有一个专用值 null，即它的字面量。值 undefined 实际上是从值 null 派生来的，因此 ECMAScript 把它们定义为相等的。
* Boolean 类型：Boolean 类型是 ECMAScript 中最常用的类型之一。它有两个值 true 和 false （即两个 Boolean 字面量）。即使 false 不等于 0，0 也可以在必要时被转换成 false，这样在 Boolean 语句中使用两者都是安全的。
* Number 类型：ECMA-262 中定义的最特殊的类型是 Number 类型。这种类型既可以表示 32 位的整数，还可以表示 64 位的浮点数。直接输入的（而不是从另一个变量访问的）任何数字都被看做 Number 类型的字面量。
* String 类型：String 类型的独特之处在于，它是唯一没有固定大小的原始类型。字符串中每个字符都有特定的位置，首字符从位置 0 开始，第二个字符在位置 1，依此类推。这意味着字符串中的最后一个字符的位置一定是字符串的长度减 1。字符串字面量是由双引号（"）或单引号（'）声明的。

对变量或值调用 typeof 运算符将返回下列值之一：

undefined - 如果变量是 Undefined 类型的

boolean - 如果变量是 Boolean 类型的

number - 如果变量是 Number 类型的

string - 如果变量是 String 类型的

object - 如果变量是一种引用类型或 Null 类型的

【引用类型】

* 引用类型通常叫做类(class)，也就是说，遇到引用值，所处理的就是对象。
* JS是基于对象而不是面向对象，对象类型的默认值是null。
* JS提供众多预定义引用类型（内置对象）

**3.运算符**

* 算数运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| + | 加 |
| - | 减 |
| \* | 乘 |
| / | 除 |
| % | 求余数 |
| ++ | 累加 |
| -- | 递减 |

* 赋值运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| = | 赋值 |
| += | 先加再赋值 |
| -= | 先减再赋值 |
| \*= | 先乘再赋值 |
| %= | 先求余数再赋值 |
| /= | 先除再赋值 |

* 比较运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| == | 判断是否等于 |
| === | 不进行数据类型转换,判断是否相等 |
| != | 判断不等于 |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |

* 逻辑运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| && | 逻辑与。两个表达式均为true则结果为true |
| || | 逻辑或。两个表达式只要有一个为true,则结果为true |
| ! | 取反。 |

**4.基本操作**

* document.getElementById()：获取页面元素
* alert()：向页面弹出提示框。
* innerHTML：操作页面某个元素的内容，可以获取，也可以赋值。
* document.write()：向页面中写内容。

## 案例分析

本案例实现注册表单的基本验证功能，主要实现非空验证、重复输入验证、邮箱验证（正则验证），通过alert提示对话框给予用户提示信息。并且当用户输入非法时阻止表单提交。

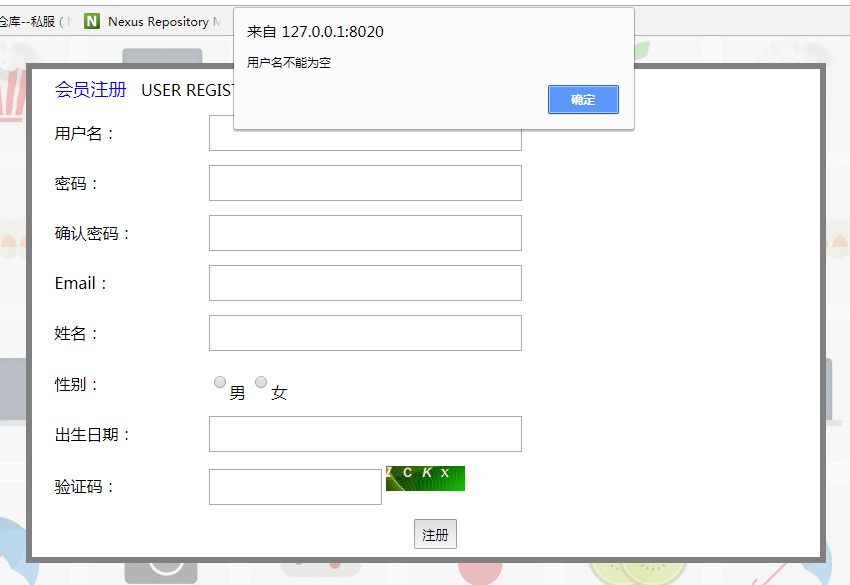
步骤分析：

* 第一步：绑定事件(onsubmit)。为form表单绑定onsubmit事件,并调用一个自定义函数。
* 第二步：编写该函数(获取用户输入的数据<获取数据时需要在指定位置定义一个 id>)
* 第三步：对用户输入的数据进行判断
* 第四步：数据合法(表单提交)
* 第五步：数据非法(给出错误提示信息，阻止表单提交)

【问题】如何控制表单提交？

关于事件 onsubmit：一般用于表单提交的位置，那么需要在定义函数的时候给出一个返回值。 onsubmit = return checkForm()

案例实现效果：当点击“注册”按钮时，验证表单输入内容是否合法，如果不合法则给出用户提示对话框，并且表单无法提交。



## 案例实现

1.步骤一：为表单注册onsubmit事件

|  |
| --- |
| <form action="#" method="get" onsubmit="return checkFrm()"> |

2.步骤二：编写checkFrm()函数

|  |
| --- |
| function checkFrm()  {  //获取用户名控件中输入的用户名  var uValue = document.getElementById("username").value;  // alert(uValue);  if(uValue=="")  {  alert("用户名不能为空");  return false;  }  var pValue = document.getElementById("userpass").value;  //验证密码非空  if(pValue=="")  {  alert("密码不能为空");  return false;  }  var pValue2 = document.getElementById("userpass2").value;  //验证两次密码一致  if(pValue!=pValue2)  {  alert("两次输入密码不一致");  return false;  }  var email = document.getElementById("email").value;  //验证邮箱是否符合规范.使用正则表达式,无需自己写,一般规则都可以通过文档找到  var rule = /^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+(.[a-zA-Z0-9\_-])+/;  if(!rule.test(email))  {  alert("输入邮箱格式非法!");  return false;  }  } |

## 案例总结

* + 1. JS中正则匹配的方式

JS中有两种匹配正则的方式：

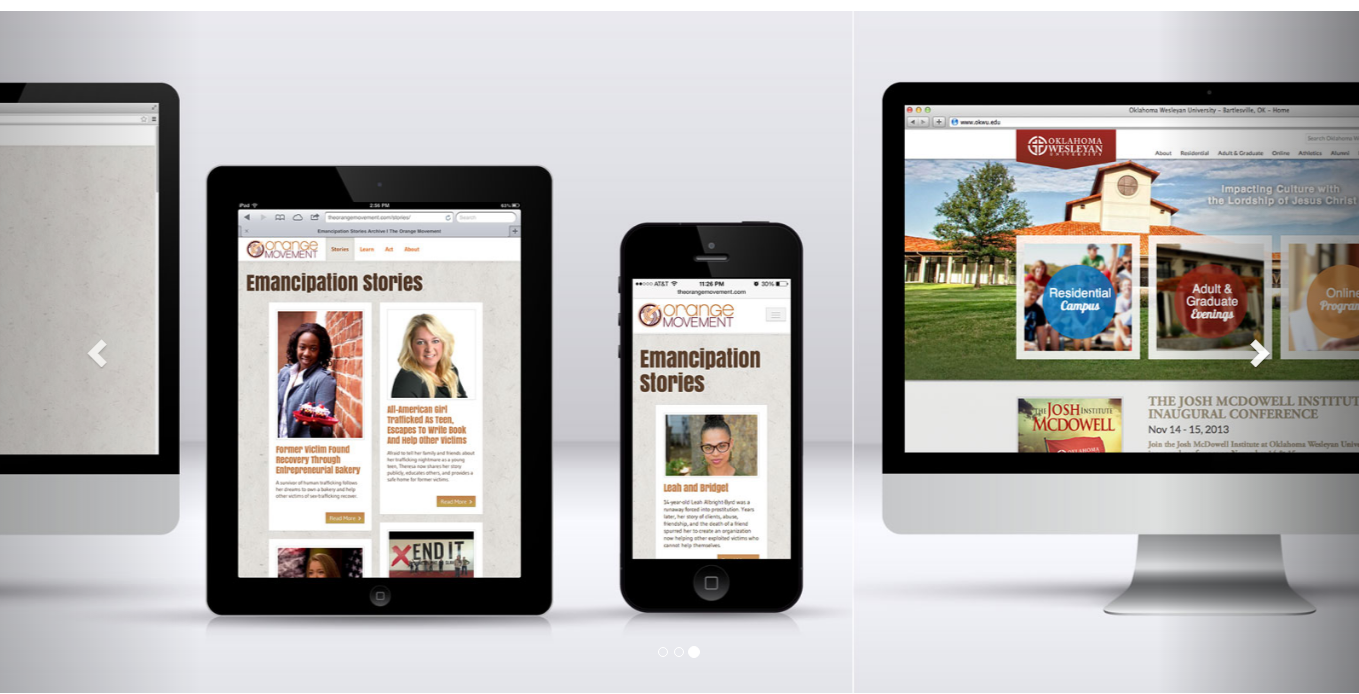
* 使用string对象中的match方法
* 使用正则对象中的test方法
  + 1. JS中的函数编写方式

函数：实现一定功能的代码块，类似于JAVA中的方法。关键字function 函数名 来自定义函数。函数可带参数，可不带参数；可带返回值，也可不带返回值。语法格式与JAVA比更加宽松。

# 案例二:完成首页轮播图功能

## 需求分析

在一天完成的首页中，轮播块只放置了一张图片显示。现在需要编写程序，完成多张图的自动切换功能。



## 相关知识点：定时器setInterval()

window.setInterval(code,millisec)按照指定的周期(间隔)来执行函数或代码段。

参数1：code必须。执行的函数名或者执行的代码段

参数2：millisec必须。间隔时间，即周期。单位毫秒。

window对象提供的都是全局函数,调用函数时可将window省略。

## 案例分析

步骤分析

第一步：确定事件(onload)并为其绑定一个函数

第二步：编写事件触发函数

第三步：编写定时任务(setInterval)

第四步：编写定时任务里面的函数

第五步：通过变量的方式，进行循环(获取轮播图的位置，并设置 src 属性)

## 代码实现

1.为轮播图片添加ID属性

|  |
| --- |
| <!--轮播-->  <div id="adv1">  <img id="adv\_img" src="../img/1.jpg" width="100%">  </div> |

2.编写函数实现三张图轮播

|  |
| --- |
| <script>  window.onload=function(){  setInterval("changeImg()",2000);  }  //自动切换3张图片  var num = 1;  function changeImg(){  if(num>3)  {  num=1;  }  //改变图片的src属性  document.getElementById("adv\_img").src="../img/"+ ++num +".jpg";  }  </script> |

# 案例三:完成定时弹出广告效果

* 1. 需求分析

我们希望在首页中的顶部做一个定时弹出广告图片。其实现效果如下：



* 1. 技术分析

获取图片的元素对象(document.getElementById(“”))

隐藏图片：display：none

定时操作：setInterval(“显示图片的函数”，3000);

清除定时操作：clearInterval(定时器标识);

* 1. 步骤分析

第一步：在页面顶部隐藏一个广告图片（使用display:none）

第二步：确定事件（onload）并为其绑定一个函数

第三步：编写这个函数（设置一个定时显示图片的操作）

第四步：编写定时器中调用的函数（显示图片 display:block）

第五步：清除显示图片的定时操作

第六步：编写函数隐藏图片的定时操作

第七步：编写定时操作中的函数（隐藏图片，使用display:none）

第八步：清除隐藏图片的定时操作

* 1. 代码实现

|  |
| --- |
| //页面加载完毕时调用init函数  window.onload=init;  //onload事件只能绑定一次。  //window.onload=adv;  function init()  {  //启动定时器,时间单位是毫秒  window.setInterval("changImg()",3000);  //2.设置图片显示的定时操作  timer = window.setInterval("showAdv()",3000);  }  //定义全局变量  var i = 1;  function changImg(){  i++;  //获取轮播图片元素  document.getElementById("adv1\_img").src="../img/"+i+".jpg";  if(i==3)  {  i=0;  }  }  //3.编写显示广告的函数  function showAdv(){  //通过设置style属性的display的值来显示广告  document.getElementById("showAdv").style.display="block";  //4.需要清除显示广告的定时操作  window.clearInterval(timer);  //5.设置隐藏广告的定时操作  timer = setInterval("hideAdv()",4000);  }  function hideAdv(){  //6.通过设置style属性的display的值来显示广告  document.getElementById("showAdv").style.display="none";  //7.需要清除显示广告的定时操作  window.clearInterval(timer);  } |

* 1. 总结

3.5.1 BOM 对象

BOM 对象：浏览器对象模型(操作与浏览器相关的内容)

* Window 对象

Window 对象表示浏览器中打开的窗口。



setInterval():它有一个返回值，主要是提供给 clearInterval 使用。

setTimeout():它有一个返回值，主要是提供给 clearTimeout 使用。

clearInterval():该方法只能清除由 setInterval 设置的定时操作

clearTimeout():该方法只能清除由 setTimeout 设置的定时操作

# 案例四:完善注册表单校验功能

* 1. 需求分析

之前我们已经使用弹出框的方式实现了表单校验的功能，但是此种方式用户体验效果极差！ 我们希望做成如下这种效果：



* 1. 技术分析

4.2.1 标签体内容:innerHTML

innerHTML指元素的内部HTML代码。只有闭合标签才有该属性，例如：div、span、h1、p等。像input标签、img标签则没有该属性。

获取:document.getElementById("id值").innerHTML

设置: document.getElementById("id值").innerHTML="赋值内容";

4.2.2 相关事件

onsubmit——提交按钮被点击

onfocus——获取焦点

onblur——失去 焦点

* 1. 步骤分析

第一步：确定事件（onfocus获取焦点）：并为其绑定一个函数。

第二步：编写这个函数：将提示信息显示在输入框后（innerHTML）

第三步：确定事件（onblur失去焦点）并为其绑定一个函数

第四步：编写该函数：实现对表单输入内容进行验证，根据验证结果给用户提示

* 1. 代码实现

1.修改HTML代码，添加错误显示区域，为文本框和密码框添加onfocus事件和onblur事件。

|  |
| --- |
| <tr>  <td width="150">用户名：</td>  <td><input type="text" name="username" size="40" id="username" onfocus="showTips('userspan','用户名必填')" onblur="checkUser()">  <span id="userspan"></span>  </td>  </tr>  <tr>  <td>密码：</td>  <td><input type="password" name="userpass" size="40" id="userpass" onfocus="showTips('passspan','密码必填')" onblur="checkPass()">  <span id="passspan"></span>  </td>  </tr> |

2.编写showTips函数和checkUser()函数

|  |
| --- |
| //显示提示信息  function showTips(id,msg){  //提示用户  document.getElementById(id).innerHTML="<font color='grey'>"+msg+"</font>";  } |

checkUser函数

|  |
| --- |
| //检查用户名  var ucheck=false;  function checkUser(){  //获取元素的value值  var uValue = document.getElementById("username").value;  //非法  if(uValue=="")  {  //给出提示  document.getElementById("userspan").innerHTML="<font color='red'>用户名不能为空</font>";  ucheck=false;  return;  }  document.getElementById("userspan").innerHTML="";  ucheck=true;  } |

3.密码校验与此类同

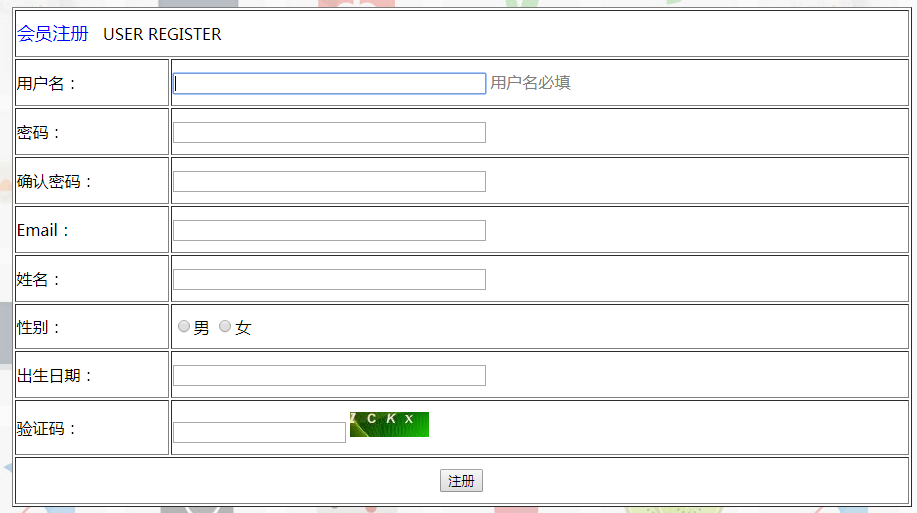
代码如下:

|  |
| --- |
| //密码验证状态  var pcheck=false;  function checkPass(){  //获取元素的value值  var pValue = document.getElementById("userpass").value;  //非法  if(pValue=="")  {  //给出提示  document.getElementById("passspan").innerHTML="<font color='red'>密码不能为空</font>";  pcheck=false;  return;  }  document.getElementById("passspan").innerHTML="";  pcheck=true;  } |

每个元素校验之后都为自己的状态赋值,只有所有状态值都为true的时候表单才能提交.因此,表单提交的事件函数处理如下:

|  |
| --- |
| //表单提交时验证函数,阻止表单的提交  function checkFrm(){  if(!ucheck||!pcheck)  {  return false;  }  return true;  } |

效果:获取焦点时



失去焦点并且非法时



通过测试，当输入有误时，表单无法提交，当输入都合法时方能提交表单。

# 总结

**内容总结:**

**1.javascript简单介绍**

1）语法

2）变量：只能使用var定义，如果在函数中定义变量，如果使用var则是局部变量，如果不使用var则就是全局变量。弱类型。

3）数据类型：原始的数据类型：undefined/null/string/number/booklean.

4）语句：  
5）运算符：== 和===

6）函数：显示函数、匿名函数

**2.BOM对象**

Window对象：alert()/setInterval()/clearInterval()/setTimeout()/clearTimeout()/confirm()/prompt()

History对象：back()/forward()/go()

Location对象：href

**3.事件**

onload:页面全部加载完毕之后执行的动作。

注意：onload只能加载一次。

onsubmit:表单提交 写在form标签中，并且使用时必须有返回值

onfucus/onblur:获取焦点/失去焦点：注册在具体的表单元素上

**4.获取页面元素**

document.getElementById("id值")

**5.获取元素值**

document.getElementById("id值").value

**6.向页面中输入内容**

alert("提示框");

document.write();

innerHTML: