HTML5表单及其验证

HTML表单一直都是Web的核心技术之一，有了它我们才能在Web上进行各种各样的应用。HTML5 Forms新增了许多新控件及其API，方便我们做更复杂的应用，而不用借助其它Javascript框架，先说下表单的几个基本知识点：

表单仍是以<form>元素作为容器，我们可在其中设置基本的提交特性；
当用户提交页面时，表单仍然向服务器发送表单控件的值；
之前老版本中的表单控件，如text radio checkbox等等，都可以按原有方式使用，尽管增加了新的功能；
仍然可以使用javascript操作表单控件。

下面详细说下HTML5中表单新增功能

1.输入型控件

Input type
用途
说明
email
电子邮件地址文本框

url
网页URL文本框

number
数值的输入域
属性 值 描述
max number 规定允许的最大值
min number 规定允许的最小值
step number 规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value number 规定默认值
range
特定值的范围的数值，以滑动条显示
属性 值 描述
max number 规定允许的最大值
min number 规定允许的最小值
step number 规定合法的数字间隔（如果 step="3"，则合法的数是 -3,0,3,6 等）
value number 规定默认值
Date pickers
日期，时间选择器
仅Opera9+支持，包含date, month, week, time, datetime, datetime-local
search
用于搜索引擎，比如在站点顶部显示的搜索框
与普通文本框用法一样，只不过这样更语文化
color
颜色选择器
仅Opera支持

将原本type为text的input控件声明为以上特殊类型，是为了给用户呈现不同的输入界面（移动平台上支持这些不同的输入界面，这里就不细说），而且表单提交时会对其值做进一步的验证。下面展示这些新表单元素，请用支持这些表单元素的浏览器查看，IE对其支持最差。
E-mail:
date:
range: number: color:

2. 表单新特性和函数

2.1 placeholder

当用户还没有输入值时，输入型控件可能通过placeholder
向用户显示描述性说明文字或者提示信息，这在目前网站中很常见，一些JS
框架都会提供类似功能，简单的说下在旧版本中常用的解决方案，为输入控件创建一个label，然后通过CSS控制些label
的位置使之覆盖在输入控件上面，当label获得焦点时，浏览器会把焦点指向输入控件。不过有了placeholder
，新的浏览器就内置了这一功能，其特性值会以浅灰色样式显示在输入框中，当输入框获得焦点并有值后，该提示信息自
动消失。
如：

```
<p><label for="runnername">Runner:</label>
        <input id="runnername"name="runnername" type="text" placeholder="First and last name" />
    </p>
```

Runner:

2.2 autocomplete

其实在IE6中,autocomplete就已经实现，不过现在这一特性终于标准化了，浏览器通过autocomplete
特性能够知晓是否应该保存输入值以备将来使用，autocomplete
应该用一保护用户敏感数据，避免本地浏览器对它们进行不安全的存储。
类型
作用
on
该字段无需保护，值可以被保存和恢复
off
该字段需要保护，值不可以保存
unspecified
包含<form>的默认设置，如果没有被包含在表单中或没有指定值，则行为表现为on
如：

```
<form action="" method="get" autocomplete="on">
Name:<input type="text" name="name" /><br />
E-mail: <input type="email" name="email" autocomplete="off" /><br />
<input type="submit" />
</form>
```

当用户提交过一次表单后，再次访问，name的输入框会提示你曾输入的值，而email则不会提示。

2.3 autofocus

页面载入时，我们通过autofocus指定某个表单元素获得焦点，但每个页面只允许出现一个autofocus
，如果设置多个则相当于未指定些行为。目前Opera10,Chromet和Safari
浏览器支持。如果用户有希望焦点转移的情况下，使用使用autofocus会惹恼用户。

2.5 list特性和datalist

通过使用list，开发人员能够为某个输入型控件构造一个选值列表，其使用方法：

```
Webpage: <input type="url" list="url_list" name="link" />
<datalist id="url_list">
    <option label="W3School" value="http://www.w3school.com.cn" />
    <option label="Google" value="http://www.google.com" />
    <option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

Webpage:
请在Opera9+或Firefox10+浏览器中查看。

2.6 required

required 属性规定必须在提交之前填写输入域（不能为空）。它是表单验证最简单的一种方式方法，使用方法：

```
Name: <input type="text" name="usr_name" required="required" />
```

2.7 pattern

pattern 属性规定用于验证 input 域的模式（pattern），模式（pattern）
是正则表达式。那些type为email或url的输入控件内置相关正则表达式，如果value
不符合其正则表达式，那表单将通不过验证，无法提交。使用方法：

Country code: <input type="text" name="country_code"
pattern="[A-z]{3}" title="Three letter country code" />

2.8 novalidate

novalidate 属性规定在提交表单时不应该验证 form 或 input 域。
如：

```
<form action="demo_form.asp" method="get" novalidate="true">
E-mail: <input type="email" name="user_email" />
<input type="submit" />
</form>
```

3. 表单验证

表单验证是一套系统，它为终端用户检测无效的数据并标记这些错误，是一种用户体验的优化，让web
应用更快的抛出错误，但它仍不能取代服务器端的验证，重要数据还要要依赖于服务器端的验证，因为前端验证是可以绕过的。
目前任何表单元素都有八种可能的验证约束条件：

名称
用途
用法
valueMissing
确保控件中的值已填写
将required属性设为true,
<input type="text"required="required"/>
typeMismatch
确保控件值与预期类型相匹配
<input type="email"/>
patternMismatch
根据pattern的正则表达式判断输入是否为合法格式
<input type="text" pattern="[0-9]{12}"/>
toolong
避免输入过多字符
设置maxLength，<textarea id="notes" name="notes" maxLength="100"></textarea>
rangeUnderflow
限制数值控件的最小值
设置min，<input type="number" min="0" value="20"/>
rangeOverflow
限制数值控件的最大值
设置max，<input type="number" max="100" value="20"/>
stepMismatch
确保输入值符合min,max,step的设置
设置max min step，<input type="number" min="0" max="100" step="10" value="20"/>
customError
处理应用代码明确设置能计算产生错误
例如验证两次输入的密码是否一致，等会DEMO细说
下面展现浏览器自带的验证功能请在Chrome、Opera或Firefox中查看：
源代码：

```
<form name="register1" id="register1">
        <p><label for="runnername">RunnerName:</label>
          <input id="runnername"name="runnername" type="text" placeholder="First and last name"
required="required" autofocus="autofocus"/>
        </p>
        <p><label for="phone">Tel #:</label>
          <input id="phone" name="phone" type="text" pattern="\d{3}-\d{4}-\d{4}"
               placeholder="xxx-xxxx-xxxx"/></p>
```

```
 <p><label for="emailaddress">E-mail:</label>
   <input id="emailaddress" name="emailaddress" type="email"
        placeholder="For confirmation only"/></p>
 <p><label for="dob">DOB:</label>
   <input id="dob" name="dob" type="date"
        placeholder="MM/DD/YYYY"/></p>
 <p>Count:<input type="number" id="count" name="count" min="0" max="100" step="10"/></p>
 <p><label for="style">Shirt style:</label>
    <input id="style" name="style" type="text" list="stylelist" title="Years of participation"
     autocomplete="off"/></p>
  <datalist id="stylelist">
   <option value="White" label="1st Year"/>
   <option value="Gray" label="2nd - 4th Year"/>
   <option value="Navy" label="Veteran (5+ Years)"/>
  </datalist>

<fieldset>
  <legend>Expectations:</legend>
  <p>
  <label for="confidence">Confidence:</label>
  <input id="confidence" name="level" type="range"
       onchange="setConfidence(this.value)"
       min="0" max="100" step="5" value="0"/>
  <span id="confidenceDisplay">0%</span></p>
  <p><label for="notes">Notes:</label>
    <textarea id="notes" name="notes" maxLength="100"></textarea></p>
</fieldset>

 <p><input type="submit" name="register" value="Submit" onclick=" checkForm()"/></p>
</form>
```

可是各个浏览器验证行为不一致，我们可能需要统一其验证行为，借助javascript我们可以统一浏览器的验证行为。
还是以上上述HTML为基础，我们为其加上相关javascript:

```
//自定义表单控件验证行为
    var checkvalue = function(e){
        var el = e.target;
        var isvalid = el.checkValidity();
        if(isvalid){
            el.className= "";
            el.parentElement.getElementsByTagName("label")[0].className="";
        }else{
            el.className= "error";
            el.parentElement.getElementsByTagName("label")[0].className="error";
        }
        e.stopPropagation();
        e.preventDefault();
    }
    //定义表单验证方法
    function invalidHandler(evt) {
        checkvalue(evt);
    }
    function loadDemo() {
```

```
            var myform = document.getElementById("register1");
            //注册表单的oninvlid事件
            myform.addEventListener("invalid", invalidHandler, true);
             for(var i=0;i< myform.elements.length-1;i++){
                //注册表单元素的onchange事件，优化用户体验
                myform.elements[i].addEventListener("change",checkvalue,false);
             }
          }
          //在页面初始化事件（onload）时注册的自定义事件
          window.addEventListener("load", loadDemo, false);
```

最后说下输入两次密码匹配的验证，写的很简单：

```
<form name="passwordChange">
   <p><label for="password1">New Password:</label>
   <input type="password" id="password1" onchange="checkPasswords()"></p>
   <p><label for="password2">Confirm Password:</label>
   <input type="password" id="password2" onchange="checkPasswords()"></p>
</form>
<button onclick="document.passwordChange.password1.checkValidity()">Check Validity</button>
function checkPasswords() {
      var pass1 = document.getElementById("password1");
      var pass2 = document.getElementById("password2");

      if (pass1.value != pass2.value)
         pass1.setCustomValidity("两次输入的密码不匹配");
      else
         pass1.setCustomValidity("");
   }
```

翻译结果