# 第三章 WEB03\_JS篇

今日任务

使用JS完成简单的数据校验(document对象、事件、函数)

使用JS完成图片轮播效果(事件、定时操作)

使用JS完成页面定时弹出广告(总结BOM对象以及JS的引入方式)

使用JS完成表单的校验(总结常用事件)

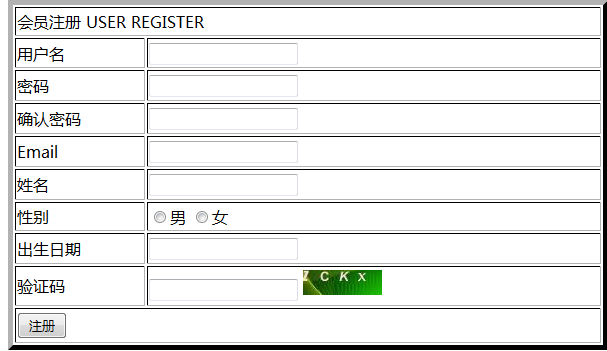
教学导航

|  |  |
| --- | --- |
| 教学目标 | 掌握JS的基本语法,数据类型,能够使用JS完成简单的页面交互.  掌握JS中的BOM对象以及JS的引入方式  掌握JS中的常用的事件  掌握JS中的常用DOM操作  了解JS中的内置对象和全局函数 |
| 教学方法 | 案例驱动法 |

## 一、使用JS完成注册表单数据校验

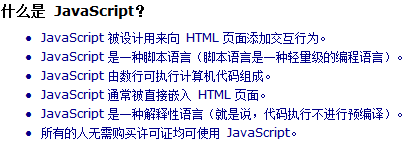
### 1.需求分析

用户在进行注册的时候会输入一些内容，但是有些用户会输入一些不合法的内容，这样会导致服务器的压力过大，此时我们需要对用户输入的内容进行一个校验(前端校验和后台校验)，前端校验防君子不防小人。



### 2.技术分析

#### 2.1 javasscript的介绍



Javascript与java没有关系(雷锋和雷峰塔)，它不是编程语言(脚本语言)

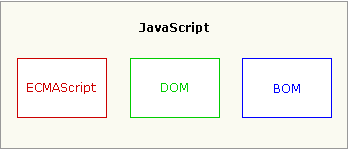
#### 2.2 javascript的作用？

HTML：它是整个网站的骨架。

CSS：它是对整个网站骨架的内容进行美化(修饰)

Javascript：它能够让整个页面具有动态效果。

#### 2.3 javascript的组成部分



ECMAScript:它是javascript的核心(语法、变量、数据类型、语句、函数……)

DOM:document object model 整个文档对象

BOM:浏览器对象

#### 2.4 javascript语法

区分大小写

变量是弱类型的(String str=”aaa” ,var str=”123”;)

每行结尾的分号可有可无(建议大家写上)

注释与java、php等语言相同。

#### 2.5javascript的变量

变量可以不用声明，变量是弱类型。统一使用var来定义！定义变量的时候不要使用关键字和保留字。

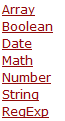
#### 2.6 javascript数据类型

Javascript数据类型分为原始数据类型和引用数据类型

原始数据类型：

string、number、boolean、null、undefined

引用数据类型：



#### 2.7 javascript运算符

其它运算符与java大体一致，需要注意其等性运算符。

== 它在做比较的时候会进行自动转换。

=== 它在做比较的时候不会进行自动转换。

#### 2.8 javascript语句

所有语句与java大体一致。

#### 2.9 获取元素内容

获取元素

document.getElementById(“id名称”);

获取元素里面的值

document.getElementById(“id名称”).value;

#### 2.10 javascript事件

表单提交事件：onsubmit

#### 2.11 javascripte的输出

警告框：alert();

向页面指定位置写入内容：innerHTML(属性)

向页面写入内容：document.write(“”);

### 3.步骤分析

第一步：确定事件(onsubmit)并为其绑定一个函数

第二步：书写这个函数(获取用户输入的数据)

第三步：对输入的数据进行判断(非空)

第四步：如果输入的内容为空，给出错误提示信息(alert),不让表单提交

第五步：如果输入的内容合法，让表单提交。

### 4.代码实现

Html部分代码：

为表单绑定一个事件

<form action="#" method="get" onsubmit="return checkForm()">

对需要校验的输入项目定义id

<input type="text" name="user" id="user" />

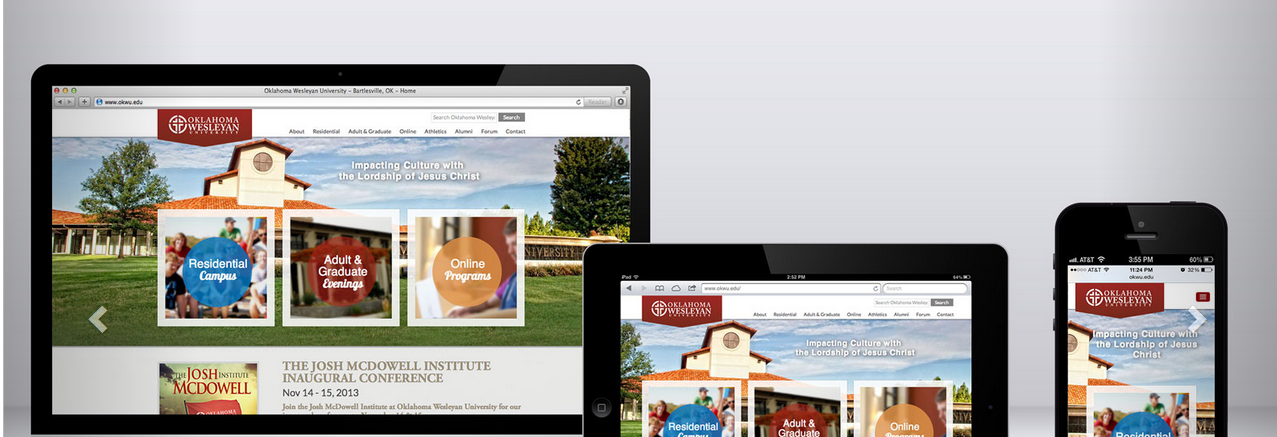
Javascript部分代码

|  |
| --- |
| <script type="text/javascript">  function checkForm(){  /\*校验用户名\*/  //alert("aa");  //获取用户输入的数据  var uValue = document.getElementById("user").value;  //alert(uValue);  if(uValue==""){  //给出错误提示信息  alert("用户名不能为空！");  return flase;  }    /\*校验邮箱\*/  var Evalue = document.getElementById("email").value;  if(!/^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+(.[a-zA-Z0-9\_-])+/.test(Evalue)){  //给出错误提示信息  alert("邮箱格式不正确！");  return false;  }  }  </script> |

## 二、使用JS完成首页轮播图效果案例

### 1.需求分析

我们希望在首页完成对轮播图的效果实现：



### 2.技术分析

获取元素 document.getElementById(“id名称”)

事件(onload)

定时操作：setInterval(“changeImg()”,3000);

### 3.步骤分析

第一步：确定事件(页面加载事件onload)并为其绑定一个函数

第二步：书写这个函数(设置一个定时操作);

第三步：书写定时操作里面的函数

第四步：获取图片的位置

第五步：为该图片设置src属性值。

### 4.代码实现

Javascript部分代码

|  |
| --- |
| <script>  function init(){  setInterval("changeImg()",3000);  }    var i=1;  function changeImg(){  i++;  var imgEl = document.getElementById("img1");  imgEl.src="../img/"+i+".jpg";  if(i==3){  i=0;  }  }    </script> |

Html部分代码

确定事件：onload 加在body里面！

<body onload="init()">

给指定的图片位置定义一个id

<img src="../img/1.jpg" width="1300px" id="img1"/>

## 三、使用JS完成页面定时弹出广告

### 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：



### 2.技术分析

获取图片的位置(document.getElementById(“”))

隐藏图片：display：none

定时操作：setInterval(“显示图片的函数”，3000);

### 3.步骤分析

第一步：确定事件(页面加载事件onload)并为其绑定一个函数init()

第二步：在init函数中设置一个显示图片的定时操作。

第三步：书写显示图片的函数(获取图片元素，设置其display属性为block)

第四步：设置一个隐藏图片的定时操作

第五步：书写隐藏图片的函数(获取图片元素，设置其display属性为none)

### 4.代码实现

Javascript部分代码：

|  |
| --- |
| <script type="text/javascript">  var time;  function init(){  //设置显示图片的定时操作  time = setInterval("showAd()",3000);  }    //显示图片的函数  function showAd(){  //获取广告图片的  var imgEl = document.getElementById("img1");  //设置图片的属性(display)让其显示  imgEl.style.display="block";    //清除显示图片的定时操作  clearInterval(time);    //设置隐藏广告图片的定时操作  time = setInterval("hiddenAd()",3000);  }    //书写隐藏广告图片的函数  function hiddenAd(){  //获取广告图片并设置隐藏的属性  document.getElementById("img1").style.display="none";  //清除隐藏图片的定时操作  clearInterval(time);  }  </script> |

Html部分代码：

<img src="../img/f001a62f-a49d-4a4d-b56f-2b6908a0002c\_g.jpg" width="1300px" style="display: none;" id="img1"/>

确定事件(<body onload="init()">)

### 5.总结

#### 5.1 javascript的引入方式

##### 内部引入方式

直接将javascript代码写到<script type=”text/javascript”></script>

##### 外部引入方式

需要创建一个.js文件，在里面书写javascript代码，然后在html文件中通过script标签的src属性引入该外部的js文件

#### 5.2 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

弹出框的几个方法：

|  |
| --- |
| <script>  //警告框  //alert("aaa");    //确认按钮  //confirm("您确认删除吗？");    //提示输入框  prompt("请输入价格：");  </script>  </head> |

##### Location对象

Location 对象包含有关当前 URL 的信息。



href:该属性可以完成通过JS代码控制页面的跳转。

|  |
| --- |
| <html>  <head>  <meta charset="UTF-8">  <title>location对象</title>  <script>  function tiao(){  window.location.href="http://www.itcast.cn";  }  </script>  </head>  <body>  <a href="#" onclick="tiao()">跳转到传智播客首页</a>  </body>  </html> |

##### History对象

History 对象包含用户（在浏览器窗口中）访问过的 URL。



历史页面：使用location页面(把href属性值改为当前的history)

History页面代码：

<input type="button" value="返回上一个页面" onclick="javascript:history.back()"/>

go(参数)

参数：-1 返回上一个历史记录页面；-2返回上上一个历史记录页面，1进入下一个历史记录页面。

让按钮点击失效：

onclick=”javascript:volid(0)”

##### Navigator对象

Navigator 对象包含有关浏览器的信息。(该对象开发中不怎么常用)



##### Screen对象

Screen 对象包含有关客户端显示屏幕的信息。(该对象开发中不怎么常用)



## 四、使用JS完成注册页面表单校验

### 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：



### 2.技术分析

使用事件(聚焦事件onfocus和离焦事件onblur),之前使用onsubmit也需要！

使用<span></span>

向页面指定位置写入内容：innerHTML属性(该属性的值存在覆盖现象)

### 3.步骤分析

第一步：确定事件(给出提示信息使用聚焦事件，给出校验结果信息使用离焦事件)并为其绑定函数

第二步：聚焦事件绑定的函数中(获取span给出提示信息)

第三步：离焦事件绑定的函数中(获取用户输入的内容进行判断)

第四步：如果失败，在span位置给出错误提示信息，如果成功，让span内容为空。

### 4.代码实现

Javascript部分代码：

|  |
| --- |
| <sctript>  function showTips(id,info){  //获取span元素，给出提示信息  document.getElementById(id+"span").innerHTML="<font color='gray'>"+info+"</font>";  }    function check(id,info){  //获取用户输入的数据  var uValue = document.getElementById(id).value;  //进行判断  if(uValue==""){  //在span位置给出错误提示信息  document.getElementById(id+"span").innerHTML="<font color='red'>"+info+"</font>";  }else{  document.getElementById(id+"span").innerHTML="";  }  }    </script> |

Html部分代码

<input type="text" name="user" id="user" onfocus="showTips('user','用户名必须以字母开头！')" onblur="check('user','用户名不能为空！')"/><span id="userspan"></span>

### 5.总结

#### 5.1 javascript的事件

我们之前已经学习过的事件(onsubmit/onload/onfocus/onblur/onclick)

其它事件：

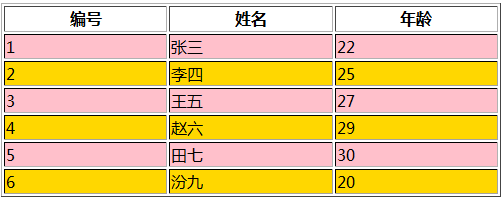


事件案例演示

## 五、使用JS完成表格的一个隔行换色

### 1.需求分析

我们希望在后台页面中实现一个隔行换色的效果显示所有的用户信息，显示效果如下：



### 2.技术分析

新标签的学习

<thead>

<tr>

<th></th>

</tr>

</thead>

<tbody>

<tr>

<td></td>

</tr>

</tbody>

确定事件(页面加载事件onload)

获取元素：获取表格(document.getElementById()),最终是为了获取表格中tbody里面的行数(长度)。

Tbody里面的行数(rows.length)

JS的遍历(for循环)

获取奇数行和偶数行(对遍历中角标对2取余)

设置背景颜色(.style.backgroundColor)

### 3.步骤分析

第一步：确定事件(onload)并为其绑定一个函数

第二步：书写函数(获取表格)

第三步：获取tbody里面的行数

第四步：对tbody里面的行进行遍历

第五步：获取奇数行和偶数行(角标对2取余)

第六步：分别对奇数行和偶数行设置背景颜色

### 4.代码实现

JS代码：

|  |
| --- |
| <script>  window.onload = function(){  //1.获取表格  var tblEle = document.getElementById("tbl");  //2.获取表格中tbody里面的行数(长度)  var len = tblEle.tBodies[0].rows.length;  //alert(len);  //3.对tbody里面的行进行遍历  for(var i=0;i<len;i++){  if(i%2==0){  //4.对偶数行设置背景颜色  tblEle.tBodies[0].rows[i].style.backgroundColor="pink";  }else{  //5.对奇数行设置背景颜色  tblEle.tBodies[0].rows[i].style.backgroundColor="gold";  }  }  }  </script> |

HTML代码：

给table里面添加一个id=“tbl”,在table里面添加新标签<thead></thead>和<tbody></tbody>

### 5.实现一个表格的高亮显示

为了加强对事件的学习，添加此案例：

分析：

第一步：确定事件(onmouseover和onmouseout)并分别为其绑定一个函数

第二步：获取鼠标移上去的那行，对其设置背景颜色

代码：

JS代码：

|  |
| --- |
| <script>  function changeColor(id,flag){  if(flag=="over"){  document.getElementById(id).style.backgroundColor="red";  }else if(flag=="out"){  document.getElementById(id).style.backgroundColor="white";  }  }  </script> |

HTML代码

<tr onmouseover="changeColor('tr1','over')" id="tr1" onmouseout="changeColor('tr1','out')">

### 6.总结

回顾之前已经使用过的事件

(onsubmit/onclick/onload/onfocus/onblur/onmouseover/onmouseout)



onfocus/onblur:聚焦离焦事件，用于表单校验的时候比较合适。

onclick/ondblclick:鼠标单击和双击事件

onkeydown/onkeypress：搜索引擎使用较多

onload：页面加载事件，所有的其它操作(匿名方式)都可以放到这个绑定的函数里面去。如果是有名称，那么在html页面中只能写一个。

onmouseover/onmouseout/onmousemove：购物网站商品详情页。

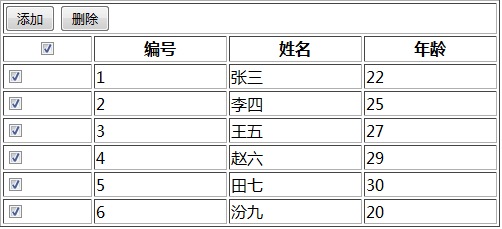
onsubmit：表单提交事件 ，有返回值，控制表单是否提交。

onchange:当用户改变内容的时候使用这个事件(二级联动)

## 六、使用JS完成全选和选不选操作

### 1.需求分析

我们希望在后台系统实现一个批量删除的操作(全选所有的复选框)，显示效果如下：



### 2.技术分析

确定事件(鼠标单击事件onclick)，事件绑定到编号前面的复选框里面

获取编号前面的复选框的状态(是否选中)

获取复选框：var checkAllEle = document.getElementById(“id”)

获取复选框的状态：checkAllEle.checked?

获取下面所有的复选框：

document.getElementsByName(“name”);

### 3.步骤分析

第一步：确定事件(onclick)并为其绑定一个函数

第二步：书写函数(获取编号前面的复选框，获取其状态)

第三步：判断编号前面复选框的状态(如果为选中，获取下面所有的复选框，并将其状态置为选中)

第四步：判断编号前面复选框的状态(如果为未选中，获取下面所有的复选框，并将其状态置为未选中)

### 4.代码实现

JS部分代码：

|  |
| --- |
| <script>  function checkAll(){  //1.获取编号前面的复选框  var checkAllEle = document.getElementById("checkAll");  //2.对编号前面复选框的状态进行判断  if(checkAllEle.checked==true){  //3.获取下面所有的复选框  var checkOnes = document.getElementsByName("checkOne");  //4.对获取的所有复选框进行遍历  for(var i=0;i<checkOnes.length;i++){  //5.拿到每一个复选框，并将其状态置为选中  checkOnes[i].checked=true;  }  }else{  //6.获取下面所有的复选框  var checkOnes = document.getElementsByName("checkOne");  //7.对获取的所有复选框进行遍历  for(var i=0;i<checkOnes.length;i++){  //8.拿到每一个复选框，并将其状态置为未选中  checkOnes[i].checked=false;  }  }  }  </script> |

HTML代码：

复选框前面的：

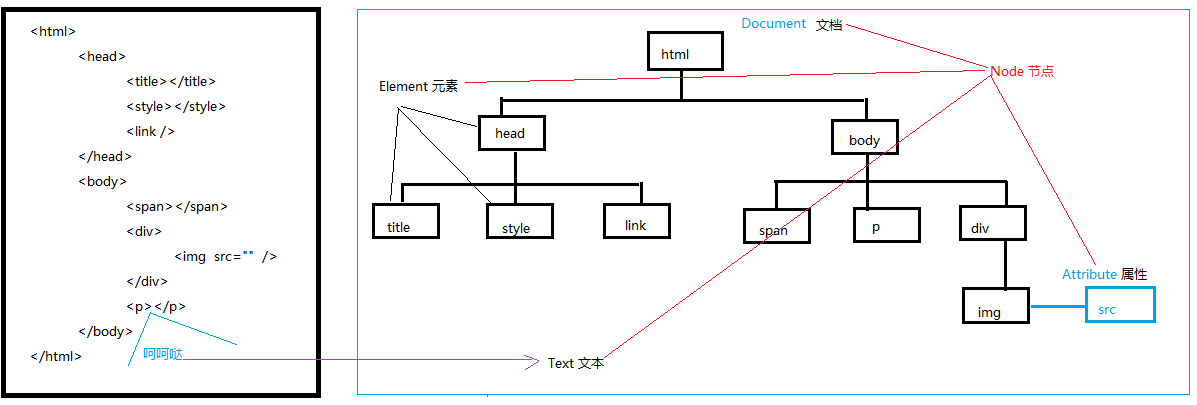
<th><input type="checkbox" onclick="checkAll()" id="checkAll"/></th>

下面所有的复选框：

<td><input type="checkbox" name="checkOne"/></td>

### 5.总结

#### 5.1 javascript的DOM操作



Document:整个html文件都成为一个document文档

Element:所有的标签都是Element元素

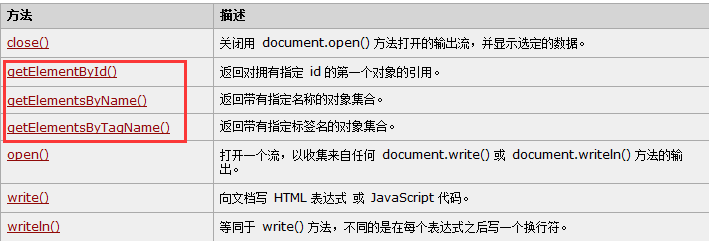
Attribute：标签里面的属性

Text：标签中间夹着的内容为text文本

Node:document、element、attribute、text统称为节点node.

##### Document对象

每个载入浏览器的 HTML 文档都会成为 Document 对象。



后面两个方法获取之后需要遍历！

以下两个方法很重要，但是在手册中查不到！

创建文本节点：document.createTextNode()

创建元素节点：document.createElement()

##### Element对象

我们所认知的html页面中所有的标签都是element元素

|  |  |
| --- | --- |
| [element.appendChild()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/met_node_appendchild.asp) | 向元素添加新的子节点，作为最后一个子节点。 |
| [element.firstChild](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/prop_node_firstchild.asp) | 返回元素的首个子节点。 |
| [element.getAttribute()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/met_element_getattribute.asp) | 返回元素节点的指定属性值。 |
| [element.innerHTML](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/prop_html_innerhtml.asp) | 设置或返回元素的内容。 |
| [element.insertBefore()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/met_node_insertbefore.asp) | 在指定的已有的子节点之前插入新节点。 |
| [element.lastChild](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/prop_node_lastchild.asp) | 返回元素的最后一个子元素。 |
| [element.setAttribute()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/met_element_setattribute.asp) | 把指定属性设置或更改为指定值。 |
| [element.removeChild()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/met_node_removechild.asp) | 从元素中移除子节点。 |
| [element.replaceChild()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/met_node_replacechild.asp) | 替换元素中的子节点。 |

##### Attribute对象

我们所认知的html页面中所有标签里面的属性都是attribute



#### 5.2 DOM练习

在页面中使用列表显示一些城市

<ul>

<li>北京</li>

<li>上海</li>

<li>广州</li>

</ul>

我们希望点击一个按钮实现动态添加城市。

分析：

事件(onclick)

获取ul元素节点

创建一个城市的文本节点

创建一个li元素节点

将文本节点添加到li元素节点中去。

使用element里面的方法appendChild()来添加子节点

代码：

JS代码

|  |
| --- |
| <script>  window.onload = function(){  document.getElementById("btn").onclick = function(){  //1.获取ul元素节点  var ulEle = document.getElementById("ul1");  //2.创建城市文本节点  var textNode = document.createTextNode("深圳");//深圳  //3.创建li元素节点  var liEle = document.createElement("li");//<li></li>  //4.将城市文本节点添加到li元素节点中去  liEle.appendChild(textNode);//<li>深圳</li>  //5.将li添加到ul中去  ulEle.appendChild(liEle);  }  }  </script> |

HTML代码

|  |
| --- |
| <input type="button" value="添加新城市" id="btn"/>  <ul id="ul1">  <li>北京</li>  <li>上海</li>  <li>广州</li>  </ul> |

## 七、使用JS完成省市二级联动

### 1.需求分析

我们希望在注册页面中添加一个字段(籍贯)，当用户选择一个具体的省份，在后面的下拉列表中动态加载该省份下所有的城市。显示的效果如下：



### 2.技术分析

事件(onchange)

使用一个二维数组来存储省份和城市(二维数组的创建？)

获取用户选择的省份(使用方法传参的方式：this.value)

遍历数组(获取省份与用户选择的省份比较，如果相同了，继续遍历该省份下所有的城市)

创建文本节点和元素节点并进行添加操作

createTextNode()

createElement()

appendChild()

### 3.步骤分析

第一步：确定事件(onchange)并为其绑定一个函数

第二步：创建一个二维数组用于存储省份和城市

第三步：获取用户选择的省份

第四步：遍历二维数组中的省份

第五步：将遍历的省份与用户选择的省份比较

第六步：如果相同，遍历该省份下所有的城市

第七步：创建城市文本节点

第八步：创建option元素节点

第九步：将城市文本节点添加到option元素节点中去

第十步：获取第二个下拉列表，并将option元素节点添加进去

第十一步：每次操作前清空第二个下拉列表的option内容。

### 4.代码实现

JS代码：

|  |
| --- |
| <script>  //1.创建一个二维数组用于存储省份和城市  var cities = new Array(3);  cities[0] = new Array("武汉市","黄冈市","襄阳市","荆州市");  cities[1] = new Array("长沙市","郴州市","株洲市","岳阳市");  cities[2] = new Array("石家庄市","邯郸市","廊坊市","保定市");  cities[3] = new Array("郑州市","洛阳市","开封市","安阳市");    function changeCity(val){    //7.获取第二个下拉列表  var cityEle = document.getElementById("city");    //9.清空第二个下拉列表的option内容  cityEle.options.length=0;    //2.遍历二维数组中的省份  for(var i=0;i<cities.length;i++){  //注意，比较的是角标  if(val==i){  //3.遍历用户选择的省份下的城市  for(var j=0;j<cities[i].length;j++){  //alert(cities[i][j]);  //4.创建城市的文本节点  var textNode = document.createTextNode(cities[i][j]);  //5.创建option元素节点  var opEle = document.createElement("option");  //6.将城市的文本节点添加到option元素节点  opEle.appendChild(textNode);  //8.将option元素节点添加到第二个下拉列表中去  cityEle.appendChild(opEle);  }  }  }  }  </script> |

Html代码：

|  |
| --- |
| <select onchange="changeCity(this.value)">  <option>--请选择--</option>  <option value="0">湖北</option>  <option value="1">湖南</option>  <option value="2">河北</option>  <option value="3">河南</option>  </select>  <select id="city">    </select> |

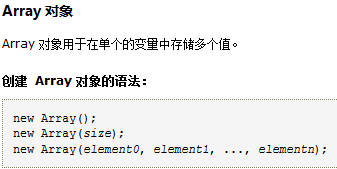
### 5.总结

#### 5.1 javascript内置对象



##### Array对象

数组的创建：

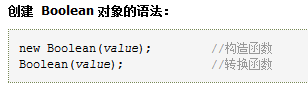


数组的特点：

长度可变！数组的长度=最大角标+1

##### Boolean对象

对象创建：



如果value 不写，那么默认创建的结果为false

##### Date对象

|  |  |
| --- | --- |
| [getTime()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/jsref_getTime.asp) | 返回 1970 年 1 月 1 日至今的毫秒数。 |

解决浏览器缓存问题

##### Math和number对象

与java里面的基本一致。

##### String对象

|  |  |
| --- | --- |
| [match()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/jsref_match.asp) | 找到一个或多个正则表达式的匹配。 |
| [substr()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/jsref_substr.asp) | 从起始索引号提取字符串中指定数目的字符。 |
| [substring()](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/jsref_substring.asp) | 提取字符串中两个指定的索引号之间的字符。 |

例子：

|  |
| --- |
| <script>  var str = "-a-b-c-d-e-f-";  var str1 = str.substr(2,4);//-b-c  //alert(str1);    var str2 = str.substring(2,4);//-b  alert(str2);  </script> |

##### RegExp对象

正则表达式对象

|  |  |
| --- | --- |
| [test](mk:@MSITStore:E:\Users\ThinkPad\Desktop\W3School离线手册(2015.07.17).chm::/www.w3school.com.cn/jsref/jsref_test_regexp.asp) | 检索字符串中指定的值。返回 true 或 false。 |

#### 5.2 全局函数

全局属性和函数可用于所有内建的 JavaScript 对象



关于编码和解码的一组方法：

|  |
| --- |
| <script>  var str = "张三";  //alert(encodeURI(str));//%E5%BC%A0%E4%B8%89  //alert(encodeURIComponent(str));//%E5%BC%A0%E4%B8%89    //alert(decodeURI(encodeURI(str)));//张三  //alert(decodeURIComponent(encodeURIComponent(str)));//张三    var str1 = "http://www.itheima.cn";  //alert(encodeURI(str1));//http://www.itheima.cn  //alert(encodeURIComponent(str1));//http%3A%2F%2Fwww.itheima.cn    //alert(decodeURI(encodeURI(str1)));//http://www.itheima.cn  //alert(decodeURIComponent(encodeURIComponent(str1)));//http://www.itheima.cn    var str2 = "alert('abc')";  //alert(str2);  eval(str2);    </script> |