day23-JavaScript高级

学习目标

1. 能够使用正则表达式进行表单的验证
2. 能够使用DOM中来查找节点
3. 能够使用DOM来增删改节点
4. 能够使用JavaScript对CSS样式进行操作
5. 能够使用数组中常用的方法
6. 能够使用日期对象常用的方法

案例一使用JS完成表单的校验plus

一,案例需求

- 用户名输入框,电子邮箱, 手机号码 失去焦点进行 校验
- 用户名：只能由英文字母和数字组成，长度为4～16个字符，并且以英文字母开头
- 手机号：以1开头, 第二为是3,4,5,7,8的11位数字　/^1[34578]\d{9}$/

二,技术分析

1.js使用正则表达式

1.1正则表达式概述

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。

用我们自己的话来说: 正则表达式用来校验字符串是否满足一定的规则的

1.2表达式的创建

- 方式一

```
var reg = new RegExp("正则表达式");
//说明：正则表达式在JS中是一个对象。Regular Expression
```

- 方式二

```
var reg = /正则表达式/;
//说明：以/开头，以/结尾，中间是正则表达式
```

- 方式一和方式二的区别

  在js中，正则表达式的两种声明方式对于“\d、\D”之类的匹配模式中，前者需要转义，而后者无需转义

  前者支持字符串拼接，支持变量，更加灵活；后者对于固定的表达式，书写起来方便快捷、更加直观。

1.3匹配模式

- 默认情况下, 正则表达式不忽略大小写的. 可以通过 i 改成忽略大小写模式

匹配模式的两种写法
var reg = new RegExp("正则表达式", "匹配模式");
var reg = /正则表达式/匹配模式;

1.4正则表达式中常用的方法

JS 中正则表达式的方法	说明
boolean test("字符串")	如果正则表达式匹配字符串，返回 true，否则返回 false

1.5常见正则表达式规则

符号	作用
\d	数字
\D	非数字
\w	单词：a-zA-Z0-9_
\W	非单词
.	通配符，匹配任意字符
{n}	匹配n次
{n,}	大于或等于n次
{n,m}	在n次和m次之间
+	1~n次
*	0~n次
?	0~1次
^	匹配开头
$	匹配结尾
[a-zA-Z]	英文字母
[a-zA-Z0-9]	英文字母和数字
[xyz]	字符集合, 匹配所包含的任意一个字符

1.6.实例代码

```
<script>
    //1.出现任意数字3次
    //a. 创建正则表达式
    var reg1  = /^\d{3}$/; //出现任意数字3次
    //b. 校验字符串
    var str1 = "3451";
    var flag1 = reg1.test(str1);
    //alert("flag1="+flag1);

    //2.只能是英文字母的, 出现6~10次之间
    var reg2  =/^[a-zA-Z]{6,10}$/;
    var str2 = "abcdef11g";
    //alert(reg2.test(str2));

    //3 用户名：只能由英文字母和数字组成，长度为4～16个字符，并且以英文字母开头
    var reg3  =/^[a-zA-Z][a-zA-Z0-9]{3,15}$/;
    var str3 = "zs";
   // alert(reg3.test(str3));


    //4. 手机号码: 以1开头, 第二位是3,4,5,6,7,8,9的11位数字
    //var reg4  =/^1[3456789]\d{9}$/i; //忽略大小写的
    var reg4  =/^1[3456789]\d{9}$/; //不忽略大小写的
    var str4 = "188245899";
    alert(reg4.test(str4));


</script>
```

2.innerHTML:向一块标签区域插入html

- 支持html标签;
- 会把之前的内容覆盖

```
<body>

    <span id="spanId">世界</span>
    <br/>
    <input type="button" value="innerHTML" onclick="writteIn()"/>

</body>
<script>
    //向span标签里面插入hello...
    // 标签对象.innerHTml = "html字符串";
    //1. 支持标签的(解析标签)
    //2. 会把之前的内容给覆盖
    function writteIn() {
        //1.获得span标签对象
        var spanEle =   document.getElementById("spanId");
        //2. 调用innerHTml属性
        spanEle.innerHTML  =  "<font color='red'>hello...</font><img src='img/gou.png'/>";
    }

</script>
```

三.思路分析

- 给用户名输入框设置获得焦点事件,

```
<input type="text" onfocus="showTips()"/>
```

- 创建一个showTips()函数响应这个事件

```
function showTips(){
  //1. 获得用户名输入框后面的span标签
  //2. 调用innerHTml属性添加提示语

}
```

- 给用户名输入框设置失去焦点事件,

```
<input type="text" onblur="checkUserName()"/>
```

- 创建一个checkUserName()函数响应这个事件

```
function checkUserName(){
  //1.获得用户输入的用户名
  //2. 使用正则表达式进行校验
  //3. 根据校验的结果, 给用户提示(span添加内容)
}
```

四,代码实现

- HTMl

```
 <tr>
   <td class="left">用户名：</td>
   <td class="center">
   <input id="usernameId" name="user" type="text" class="in" onfocus="showTips()"
onblur="checkUserName(this)" />
   <span id="usernamespan"></span>
   </td>
 </tr>
```

- JS代码

```
    <script type="text/javascript">
            //创建一个showTips()函数响应获得焦点事件
            function showTips() {
                //1. 获得用户名输入框后面的span标签
                var spanEle =  document.getElementById("usernamespan");
                //2. 调用innerHTMl属性添加提示语
                spanEle.innerHTML  ="由英文字母和数字组成(4～16位)，并且以英文字母开头";
            }

            //创建一个checkUserName()函数响应失去焦点事件
            function checkUserName(obj) {
                //1.获得用户输入的用户名
                var username =  obj.value;
                //2. 使用正则表达式进行校验
                var reg  =/^[a-zA-Z][a-zA-Z0-9]{3,15}$/;
                var flag =  reg.test(username);
                //3. 根据校验的结果，给用户提示(span添加内容)
                if (flag){
                    //符合规则
                    document.getElementById("usernamespan").innerHTML = "<img src='img/gou.png'
width='30' height='20px'/>";

                }else{
                    //不合符规则
                    document.getElementById("usernamespan").innerHTML ="用户名不合法";

                }
            }

        </script>
```

案例二:使用JS控制复选框的全选和全不选的效果

一,需求分析

	商品名称	商品价格	商品数量	操作
	MAC	18000	10	删除 修改
	山地自行车	1800	100	删除 修改
	iPhonex	9000	100	删除 修改
	苹果	9	100	删除 修改

- 最上面的复选框被选中,下面的全部也被选中; 最上面的复选框不选中,下面的全部不选中;

二,技术分析

1.DOM概念

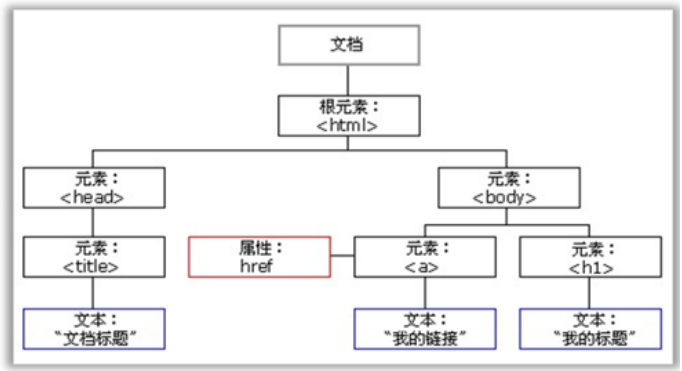
文档对象模型（Document Object Model，简称DOM），可以让用户对网页中的元素(标签)进行操作

2.JS里面的DOM编程

每个HTML页面在被浏览器解析的时候都会在内存中创建一棵DOM树，我们通过编写JS代码就可以访问这棵树上任何一个节点，并且对节点进行操作。通过 DOM模型，可以访问所有的 HTML 元素，连同它们所包含的文本和属性。可以对其中的内容进行修改和删除，同时也可以创建新的元素。新创建的元素对象，要挂到DOM树上才可以在网页上显示出来。

```
<html>
  <head>
    <title>文档标题</title>
  </head>
  <body>
    <a href="">我的链接</a>
    <h1>我的标题</h1>
  </body>
</html>
```

DOM 树



整个文档是一个文档节点

每个 HTML 元素(标签)是元素(标签)节点

HTML 元素内的文本是文本节点

每个 标签 属性是属性节点

注释是注释节点

- 一切皆节点,一切皆对象

4.dom定义的方法

方法	描述
getElementById()	返回带有指定 ID 的元素。
getElementsByTagName()	返回包含带有指定标签名称的所有元素的节点列表（集合/节点数组）。
getElementsByClassName()	返回包含带有指定类名的所有元素的节点列表。
appendChild()	把新的子节点添加到指定节点。
removeChild()	删除子节点。
replaceChild()	替换子节点。
insertBefore()	在指定的子节点前面插入新的子节点。
createAttribute()	创建属性节点。
createElement()	创建元素节点。
createTextNode()	创建文本节点。
getAttribute()	返回指定的属性值。
setAttribute()	把指定属性设置或修改为指定的值。

4.1查找节点的方法

获取元素的方法	作用
document.getElementById("id")	通过 id 属性找到唯一的元素 如果页面上有多个同名的 id，则得到第 1 个元素
document.getElementsByName("name")	通过 name 属性得到一组标签，返回一个数组
document.getElementsByTagName ("标签名")	通过标签名字得到一组标签，返回一个数组。如：input
document.getElementsByClassName("类名")	通过类名得到一组标签，返回一个数组

- 示例代码

```
<script>
    //1. 根据id来获得标签节点(对象)
    var inputEle =  document.getElementById("inputId");

    //2. 根据标签名获得标签节点数组
    var inputEles =  document.getElementsByTagName("input");
    //console.log("input标签的数量="+inputEles.length);
    for(var i = 0; i < inputEles.length;i++){
        inputEles[i];
    }

    //3.根据类名来获得标签节点数组
    var inputEles2  = document.getElementsByClassName("inputClass");
    console.log(inputEles2.length);

</script>
```

4.2增删改节点

在DOM树上创建元素分2步：

1. 创建这个元素
2. 把元素挂到DOM树上

4.2.1创建和修改元素的方法

创建元素的方法	作用
document.createElement("标签名")	在文档上创建一个元素对象
元素对象.setAttribute("属性名", "属性值")	给元素添加一个属性名和属性值 如果属性名不存在则是添加属性，存在则是修改属性值
document.createTextNode("文本内容")	在文档上创建一个文本节点

4.2.2挂到DOM树的方法

将元素挂到 DOM 树上的方法	作用
父元素.appendChild(子元素)	将元素追加成父元素的最后一个子元素
父元素.removeChild(子元素)	通过父元素删除一个子元素
元素.remove()	元素删除本身

三,思路分析

- 创建这个页面
- 给最上面的复选框设置点击事件
- 创建selectAll()的函数响应这个事件
- 在selectAll()的函数里面:

    //1. 获得下面四个复选框对象数组

    //2. 遍历这个数组,

        判断最上面的复选框是否选中(checked属性),

        如果选中, 下面的复选框也选中

        如果不选中, 下面的复选框也不选中

四,代码实现

```
<script>
    function  selectAll(obj) {
        //1. 获得下面四个复选框对象数组
        var ones  = document.getElementsByClassName("one");
        //2.  遍历这个数组,
        for(var i = 0; i < ones.length;i++){
            // 判断最上面的复选框是否选中(checked属性),
            if (obj.checked){
                //如果选中, 下面的复选框也选中
                ones[i].checked = true;
            }else{
                //如果不选中, 下面的复选框也不选中
                ones[i].checked = false;
            }
        }
    }

</script>
```

五,扩展_案例二优化

```
<script>
    function  selectAll(obj) {
        //1. 获得下面四个复选框对象数组
        var ones  = document.getElementsByClassName("one");

        //获得最上面复选框状态
        var flag  =obj.checked;


        //2.  遍历这个数组,
        for(var i = 0; i < ones.length;i++){
            // 下面的复选框的状态和最上面的复选框状态一致
            ones[i].checked = flag;
        }
    }

</script>
```

案例三:使用JS控制表格的隔行换色

一,需求分析

	商品名称	商品价格	商品数量	操作
	MAC	18000	10	删除 修改
	MAC	18000	10	删除 修改
	MAC	18000	10	删除 修改
	MAC	18000	10	删除 修改

- 使用JS修改表格行的背景色，产生隔行变色的效果.

二,技术分析

1.使用JS操作CSS样式

1.1 在JS中操作CSS属性命名上的区别

CSS 中写法	JS 中的写法	说明
color	color	如果一个单词写法一样
font-size	fontSize	如果多个单词，使用驼峰命名法

1.2操作CSS样式

- 方式一

```
标签对象.style.样式名 = "样式值"
注: 每条语句只能修改一个样式
```

- 方式二

```
元素对象.className = "样式类名"
注:每条语句修改一批样式
```

三,思路分析

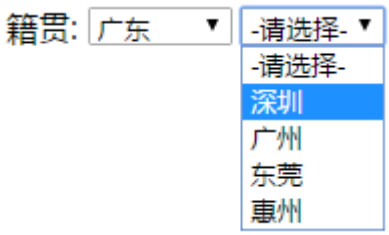
- 创建页面
- 获得所有的行标签数组 document.getElementsByTagName();
- 遍历行标签数组, 判断是奇数行还是偶数行, 给奇数行设置一个背景色, 给偶数行也设置一个另外背景色

四,代码实现

```
<script>
    //1. 获得所有的行标签数组
    var trELes =  document.getElementsByTagName("tr");

    //2. 遍历行标签数组,
    for(var i  = 0; i < trELes.length; i++){
        //判断是奇数行还是偶数行,
        if (i % 2 == 0){
            // 给奇数行设置一个背景色,
            trELes[i].style.backgroundColor = "#4F81BD";
        }else{
            // 给偶数行也设置一个另外背景色
            trELes[i].style.backgroundColor = "#FFB6C1";
        }

    }


</script>
```

案例四:JS控制二级联动

一,需求分析



- 在注册页面添加籍贯,左边是省份的下拉列表,右边是城市的下拉列表.右边的select根据左边的改变而更新数据

二,技术分析

1.js内置对象之数组

1. 创建数组的方式

创建数组的方式	说明
new Array()	创建 0 个长度的数组
new Array(5)	创建 5 个长度的数组
new Array(2,4,10,6,41)	指定每个元素创建数组
[4,3,20,6]	使用中括号指定每个元素

2. JS数组的特点

- 数组里面可以放不同类型的值(和Java不一样)
- 数组的长度是可变的
- 数组中是有方法的

方法名	功能
concat()	连接两个或更多的数组，并返回结果
reverse()	将数组进行反转
join(separator)	与 split()功能相反，将数组通过分隔符，拼成一个字符串。

```
<script>
    /**
     * 数组的特点:
     * 1. 数组里面可以存放不同类型的数据
     * 2. 数组没有越界概念的, 长度可变的
     * 3. 数组是有方法的
     */

    //定义了一个长度为5的数组, 并且赋值了

    var array = [1,2,3,4,"哈哈哈"];

    console.log(array.length); //5

   // console.log(array[0]); //1
   // console.log(array[4]); //哈哈哈

    array[8] = 8888; // var array = [1,2,3,4,"哈哈哈",undefined,undefined,undefined,8888]

    console.log(array.length);// 9


    //console.log(array[8]);//在java里面数组越界, 在js里面undefined


    /*for(var i = 0; i <  array.length;i++){
        console.log("array["+i+"]="+array[i]);
    }*/



</script>
```

3. 二维数组

```
<script>
    //定义了一个长度为3的数组
    var citys = new Array(3);

    citys[0] = ["深圳","惠州","东莞","广州"];
    citys[1] = ["武汉","黄冈","鄂州","黄石"];
    citys[2] = ["济南","青岛","威海","日照","烟台"];


    //遍历
    for(var i = 0; i < citys.length;i++){
        var cityArray =  citys[i];
        console.log("cityArray="+cityArray);

        for(var j = 0; j  < cityArray.length;j++){
            console.log("city="+cityArray[j]);
        }

    }

    var citys02 = [["深圳","惠州","东莞","广州"],["武汉","黄冈","鄂州","黄石"],["济南","青岛","威海","日照","烟台"]];

</script>
```

2.js内置对象之date

1. 日期对象的创建

```
var 变量名 =new Date()  // Date 对象会自动把当前日期和时间保存为其初始值。
```

2. 日期对象常见的方法

方法名	作用
getFullYear()	从 Date 对象以四位数字返回年份。
getMonth()	从 Date 对象返回月份 (0 ~ 11)。
getDate()	从 Date 对象返回一个月中的某一天 (1 ~ 31)。
getDay()	从 Date 对象返回一周中的某一天 (0 ~ 6)。其中：0 表示周日，1~6 周一到周六
getHours()	返回 Date 对象的小时 (0 ~ 23)。
getMinutes()	返回 Date 对象的分钟 (0 ~ 59)。
getSeconds()	返回 Date 对象的秒数 (0 ~ 59)。
getMilliseconds()	返回 Date 对象的毫秒(0 ~ 999)。
getTime()	返回 1970 年 1 月 1 日至今的毫秒数。类似于 Java 中的 System.currentTimeMillis()
toLocaleString()	根据本地时间格式，把 Date 对象转换为字符串。

三,思路分析

- 创建这个页面 (两个select)
- 给省份的select设置一个内容改变事件

```
<select onchange="refreshCity()"></select>
```

- 创建refreshCity()的函数响应这个事件

```
function refreshCity(){
  //1. 获得选择的省份的值
  //2. 根据省份的值 获得当前省份下面的城市的数据  ["深圳","惠州","东莞","广州"]
  //3. 遍历城市的数据(数组)
  //4. 把每一个参数, 创建成 <option>城市的值(深圳)</option>
  //5. 把创建好的optio添加到城市的select里面

}
```

四,代码实现

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    籍贯: <select id="provinceId" onchange="refreshCity(this)">
                <option value="-1">-请选择-</option>
                <option value="0">广东</option>
                <option value="1">湖北</option>
                <option value="2">山东</option>
          </select>
          <select id="cityId">
              <option>-请选择-</option>
          </select>

</body>

<script>
    //初始化数据
    var citys = new Array(3);
    citys[0] = ["深圳","惠州","东莞","广州"];
    citys[1] = ["武汉","黄冈","鄂州","黄石"];
    citys[2] = ["济南","青岛","威海","日照","烟台"];

    //创建refreshCity()的函数响应省份的slect内容改变事件
    function refreshCity(obj) {
        //0 获得城市的select
        var citySelect =  document.getElementById("cityId");

        // 清空城市的select里面之前的option innerHTML;
        citySelect.innerHTML  = "<option>-请选择-</option>";

        //1. 获得选择的省份的值
        var pValue =  obj.value;
        //2. 根据省份的值 获得当前省份下面的城市的数据  ["深圳","惠州","东莞","广州"]
        if (pValue > -1){ //eg: 如果选择的是广东, pValue = 0
           var cityArray =   citys[pValue];//["深圳","惠州","东莞","广州"]
            //3. 遍历城市的数据(数组)
            for(var i = 0; i < cityArray.length;i++){
                //4. 把每一个城市, 创建成 <option>深圳</option>
                var cityValue =  cityArray[i]; //获得每一个城市的数据 eg: 深圳
                //a . 创建option节点 <option></option>
                 var opEle =   document.createElement("option");
                 //b 创建文本节点  深圳
                 var textNode =   document.createTextNode(cityValue);
                 //c. 把文本节点添加到option节点  <option>深圳</option>
                opEle.appendChild(textNode);
                //5. 把创建好的optio添加到城市的select里面
                citySelect.appendChild(opEle);


            }
```

```
        }

    }


</script>
</html>
```

扩展案例_ 电子时钟

一, 案例需求

2018/07/30 下午18:08:33

- 每隔1s 向页面打印一次 系统时间

二, 技术分析

- 定时任务

```
setInterval(code, time);
```

- 日期对象
- innerHTML

三, 思路分析

```
<span id="spanId">  </span>


 setInterval("showTime()",1000);

 function showTime(){
     //1. 获得时间
     //2. 把时间插入到span里面 (innerHTML)

 }
```