


# JavaScript Demo

## 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出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。

会员注册 USER REGISTER	
用户名	<input type="text"/>
密码	<input type="password"/>
确认密码	<input type="password"/>
Email	<input type="text"/>
姓名	<input type="text"/>
性别	<input type="radio"/> 男 <input type="radio"/> 女
出生日期	<input type="text"/>
验证码	<input type="text"/> 
<input type="button" value="注册"/>	

### 2、技术分析

#### 2.1 JavaScript 的介绍

##### 什么是 JavaScript？

- JavaScript 被设计用来向 HTML 页面添加交互行为。
- JavaScript 是一种脚本语言（脚本语言是一种轻量级的编程语言）。

- JavaScript 由数行可执行计算机代码组成。
- JavaScript 通常被直接嵌入 HTML 页面。
- JavaScript 是一种解释性语言（就是说，代码执行不进行预编译）。
- 所有的人无需购买许可证均可使用 JavaScript。

Java 与 JavaScript 没有关系。

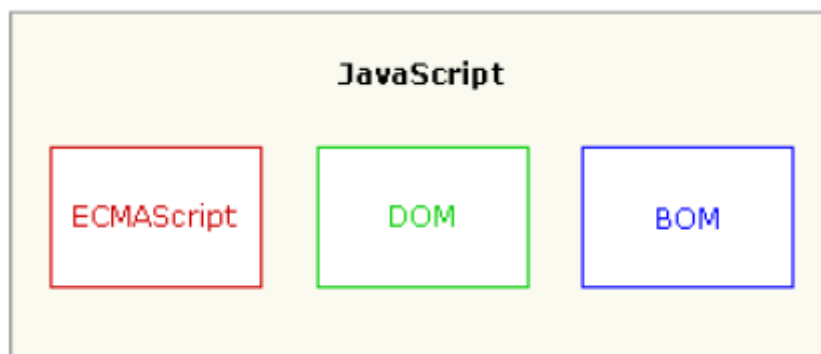
## 2.2 JavaScript 的作用

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：它是整个 JavaScript 的核心，包含（基本语法、变量、关键字、保留字、数据类型、语句、函数等等）。

DOM：文档对象模型，包含（整个 html 页面的内容）。

BOM：浏览器对象模型，包含（整个浏览器相关内容）。

## 2.4 JavaScript 语法

区分大小写；

变量是弱类型（`String str="aaa", var str="123";`）；

每行结尾的分号可有可无（建议写上分号）；

变量不一定要初始化，声明变量不是必须的

注释与 Java、php 等语言相同。

## 2.5 JavaScript 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：Array、Boolean、Date、Math、Number、String、RegExp。

## 2.7 JavaScript 运算符

其他运算符与 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。

=== 它在作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")； **批注**：如果 id 是一个字符串，那么必须加上引号，如果是一个变量那么不需要。

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 JavaScript 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<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 ( 让表单提交 );

第五步：数据非法 ( 给出错误提示信息，不让表单提交 );

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()`

## 4、代码实现

JS 代码：

```
<script>
function checkForm(){
    //alert("aa");
}
```

```

    /**校验用户名**/

    //1、获取用户输入的数据
    var uValue = document.getElementById("user").value;
    //alert(uValue);
    if(uValue==""){

        //2、给出错误提示信息

        alert("用户名不能为空!");

        return false;
    }

    //校验密码

    var pValue = document.getElementById("password").value;
    if(pValue==""){

        alert("密码不能为空!");

        return false;
    }

    //校验确认密码

    var rpValue = document.getElementById("repassword").value;
    if(rpValue==""){

        alert("两次密码输入不一致!");

        return false;
    }

    /*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)){

        alert("邮箱格式不正确!");

        return false;
    }
}
</script>

```

HTML 部分：

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

## 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 ( `onload` ) 并为其绑定一个函数；

第二步：书写绑定的这个函数

第三步：书写定时任务 ( `setInterval` )

第四步：书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 )；**批注**：在循环的时候需要注意，到了最后一张图片的时候要重置。

## 4、代码实现

JS 代码：

```
<script>
    function init(){

        //书写轮图显示的定时操作

        window.setInterval("changeImg()",3000);

    }

    //书写函数

    var i=0;
    function changeImg(){
        i++;

        //获取图片位置并设置 src 属性值

        document.getElementById("img1").src="../img/"+i+".jpg";
        if(i==3){
            i=0;
        }
    }
}
</script>
```

HTML：

```
<body onload="init()">
```

在指定位置定义 id。

## 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：在页面指定位置隐藏一个广告图片 ( 使用 display 属性的 none 值 )

第二步：确定事件 ( onload ) 并为其绑定一个函数

第三步：书写这个函数 ( 设置一个显示图片的定时操作 )

第四步：书写定时器中的函数 ( 获取广告图片的位置并设置属性 style 的 display 值 block )

第五步：清除显示图片的定时操作 ( )

第六步：书写隐藏图片的定时操作

第七步：书写定时器中的函数 ( 获取广告图片的位置并设置属性 style 的 display 值 none )

第八步：清除隐藏图片的定时操作 ( )

## 4、代码实现

JS 代码：

```
<script>
    function init(){

        //书写轮图片显示的定时操作

        window.setInterval("changeImg()",3000);

        //1、设置显示广告图片的定时操作

        time = window.setInterval("showAd()",3000);
    }

    //书写函数

    var i=0;
```



```

function changeImg(){
    i++;

    //获取图片位置并设置 src 属性值

    document.getElementById("img1").src="../img/"+i+".jpg";
    if(i==3){
        i=0;
    }
}

//2、书写显示广告图片的函数

function showAd(){

    //3、获取广告图片的位置

    var adEle = document.getElementById("img2");

    //4、修改广告图片元素里面的属性让其显示

    adEle.style.display = "block";

    //5、清除显示图片的定时操作

    clearInterval(time);

    //6、设置隐藏图片的定时操作

    setInterval("hiddenAd()",3000);
}

//7、书写隐藏广告图片的函数

function hiddenAd(){

    //8、获取广告图片并设置其 style 属性的 display 值为 none

    document.getElementById("img2").style.display= "none";

    //9、清除隐藏广告图片的定时操作

    clearInterval(time);
}
</script>

```

HTML 代码：

```

<body onload="init()">

<!--定时弹出广告图片位置-->



```

## 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。

## Window 对象方法

方法	描述
<a href="#">alert()</a>	显示带有一段消息和一个确认按钮的警告框。
<a href="#">blur()</a>	把键盘焦点从顶层窗口移开。
<a href="#">clearInterval()</a>	取消由 <a href="#">setInterval()</a> 设置的 <code>timeout</code> 。
<a href="#">clearTimeout()</a>	取消由 <a href="#">setTimeout()</a> 方法设置的 <code>timeout</code> 。
<a href="#">close()</a>	关闭浏览器窗口。
<a href="#">confirm()</a>	显示带有一段消息以及确认按钮和取消按钮的对话框。
<a href="#">createPopup()</a>	创建一个 <code>pop-up</code> 窗口。
<a href="#">focus()</a>	把键盘焦点给予一个窗口。
<a href="#">moveBy()</a>	可相对窗口的当前坐标把它移动指定的像素。
<a href="#">moveTo()</a>	把窗口的左上角移动到一个指定的坐标。
<a href="#">open()</a>	打开一个新的浏览器窗口或查找一个已命名的窗口。
<a href="#">print()</a>	打印当前窗口的内容。
<a href="#">prompt()</a>	显示可提示用户输入的对话框。
<a href="#">resizeBy()</a>	按照指定的像素调整窗口的大小。
<a href="#">resizeTo()</a>	把窗口的大小调整到指定的宽度和高度。
<a href="#">scrollBy()</a>	按照指定的像素值来滚动内容。
<a href="#">scrollTo()</a>	把内容滚动到指定的坐标。
<a href="#">setInterval()</a>	按照指定的周期（以毫秒计）来调用函数或计算表达式。
<a href="#">setTimeout()</a>	在指定的毫秒数后调用函数或计算表达式。

`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：

```
<head>
<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。

### Location 对象属性

属性	描述
<a href="#">hash</a>	设置或返回从井号 (#) 开始的 URL (锚)。
<a href="#">host</a>	设置或返回主机名和当前 URL 的端口号。
<a href="#">hostname</a>	设置或返回当前 URL 的主机名。
<a href="#">href</a>	设置或返回完整的 URL。
<a href="#">pathname</a>	设置或返回当前 URL 的路径部分。
<a href="#">port</a>	设置或返回当前 URL 的端口号。
<a href="#">protocol</a>	设置或返回当前 URL 的协议。
<a href="#">search</a>	设置或返回从问号 (?) 开始的 URL (查询部分)。

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=www.baidu.com;
      }
    </script>
  </head>
  <body>
```

```
<a href="#" onclick="tiao()">跳转到百度首页</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。

### History 对象方法

方法	描述
<u><a href="#">back()</a></u>	加载 history 列表中的前一个 URL。
<u><a href="#">forward()</a></u>	加载 history 列表中的下一个 URL。
<u><a href="#">go()</a></u>	加载 history 列表中的某个具体页面。

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:void(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)

### Navigator 对象属性

属性	描述
<a href="#">appName</a>	返回浏览器的代码名。
<a href="#">appMinorVersion</a>	返回浏览器的次级版本。
<a href="#">appName</a>	返回浏览器的名称。
<a href="#">appVersion</a>	返回浏览器的平台和版本信息。
<a href="#">browserLanguage</a>	返回当前浏览器的语言。
<a href="#">cookieEnabled</a>	返回指明浏览器中是否启用 cookie 的布尔值。
<a href="#">cpuClass</a>	返回浏览器系统的 CPU 等级。
<a href="#">onLine</a>	返回指明系统是否处于脱机模式的布尔值。
<a href="#">platform</a>	返回运行浏览器的操作系统平台。
<a href="#">systemLanguage</a>	返回 OS 使用的默认语言。
<a href="#">userAgent</a>	返回由客户机发送服务器的 user-agent 头部的值。
<a href="#">userLanguage</a>	返回 OS 的自然语言设置。

# 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)

## Screen 对象属性

属性	描述
<a href="#">availHeight</a>	返回显示屏幕的高度 (除 Windows 任务栏之外)。
<a href="#">availWidth</a>	返回显示屏幕的宽度 (除 Windows 任务栏之外)。
<a href="#">bufferDepth</a>	设置或返回调色板的比特深度。
<a href="#">colorDepth</a>	返回目标设备或缓冲器上的调色板的比特深度。
<a href="#">deviceXDPI</a>	返回显示屏幕的每英寸水平点数。
<a href="#">deviceYDPI</a>	返回显示屏幕的每英寸垂直点数。
<a href="#">fontSmoothingEnabled</a>	返回用户是否在显示控制面板中启用了字体平滑。
<a href="#">height</a>	返回显示屏幕的高度。
<a href="#">logicalXDPI</a>	返回显示屏幕每英寸的水平方向的常规点数。
<a href="#">logicalYDPI</a>	返回显示屏幕每英寸的垂直方向的常规点数。
<a href="#">pixelDepth</a>	返回显示屏幕的颜色分辨率 (比特每像素)。
<a href="#">updateInterval</a>	设置或返回屏幕的刷新率。
<a href="#">width</a>	返回显示器屏幕的宽度。

## 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：

会员注册 USER REGISTER	
用户名	<input type="text"/> 提示信息和校验结果显示在该位置
密码	<input type="password"/>
确认密码	<input type="password"/>
Email	<input type="text"/>
姓名	<input type="text"/>
性别	<input type="radio"/> 男 <input type="radio"/> 女
出生日期	<input type="text"/>
验证码	<input type="text"/> 
<input type="button" value="注册"/>	

## 2、技术分析

## 3、步骤分析

第一步：确定事件（onfocus 聚焦事件）并为其绑定一个函数

第二步：书写绑定函数（在输入框的后面给出提示信息）

第三步：确定事件（onblur 离焦事件）并为其绑定一个函数

第四步：书写函数（对数据进行校验，分别给出提示）

## 4、代码实现

HTML 代码

```
<input type="text" name="user" size="34px" id="user" onfocus="showTips('user','用户名必填!')" onblur="check('user','用户名不能为空!')"><span id="userspan"></span>
```

JS 代码：

```
function showTips(id,info){
    document.getElementById(id+"span").innerHTML="<font color='gray'>" + info + "</font>";
}
```



```
function check(id,info){  
    //1、获取用户输入的用户名数据  
    var uValue = document.getElementById(id).value;  
    //2、进行校验  
    if(uValue==""){  
        document.getElementById(id+"span").innerHTML="<font  
color='red'>" + info + "</font>"  
    }else{  
        document.getElementById(id+"span").innerHTML="";  
    }  
}
```

# javascript 简单介绍

## ECMAScript

1. 语法
2. 变量：只能使用 var 定义，如果在函数的内容使用 var 定义，那么它是一个局部变量，如果没有使用 var，那么它是一个全局的，弱类型！
3. 数据类型：原始数据类型（undefined/null/string/boolean）
4. 语句：
5. 运算符：==与===
6. 函数：两种写法（有名称，匿名的）

## BOM 对象

window:alert(),prompt(),confirm(),setInterval(),clearInterval(),setTimeout(),clearTimeout()

history:go(参数), back(),forward()

location:href 属性

## 事件

onsubmit() 此事件写在 form 标签中，必须有返回值。

onload() 此事件只能写一次并且放到 body 标签中

其他事件放到需要操作的元素位置。(onclick、onfocus、onblur)

## 获取元素

```
document.getElementById("id")
```

## 获取元素里面的值

```
document.getElementById("id").value
```

## 向页面输出

弹窗：alert();

向浏览器中写入内容：document.write(内容);

向页面指定位置写入内容：innerHTML

# 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：

编号	姓名	年龄
1	张三	22
2	李四	25
3	王五	27
4	赵六	29
5	田七	30
6	汾九	20

## 2、技术分析

### 新标签的学习

```
<thead>
  <tr>
    <td></td>
  </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>
    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')">

<tr onmouseover="changeColor('tr2','over')" id="tr2"
onmouseout="changeColor('tr2','out')">
```

## 5、实现一个表格的高亮显示

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，对其设置背景色

## 6、总结

回顾之前已经使用过的事件

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

属性	当以下情况发生时，出现此事件	FF	N	IE
onabort	图像加载被中断	1	3	4
onblur	元素失去焦点	1	2	3
onchange	用户改变域的内容	1	2	3
onclick	鼠标点击某个对象	1	2	3
ondblclick	鼠标双击某个对象	1	4	4
onerror	当加载文档或图像时发生某个错误	1	3	4
onfocus	元素获得焦点	1	2	3
onkeydown	某个键盘的键被按下	1	4	3
onkeypress	某个键盘的键被按下或按住	1	4	3
onkeyup	某个键盘的键被松开	1	4	3
onload	某个页面或图像被完成加载	1	2	3
onmousedown	某个鼠标按键被按下	1	4	4
onmousemove	鼠标被移动	1	6	3
onmouseout	鼠标从某元素移开	1	4	4
onmouseover	鼠标被移到某元素之上	1	2	3
onmouseup	某个鼠标按键被松开	1	4	4
onreset	重置按钮被点击	1	3	4
onresize	窗口或框架被调整尺寸	1	4	4
onselect	文本被选定	1	2	3
onsubmit	提交按钮被点击	1	2	3
onunload	用户退出页面	1	2	3

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，所有的其他操作（匿名方式）都可以放到这个绑定的函数里面去。

如果有名称，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

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：

<input type="button" value="添加"/> <input type="button" value="删除"/>			
<input checked="" type="checkbox"/>	编号	姓名	年龄
<input checked="" type="checkbox"/>	1	张三	22
<input checked="" type="checkbox"/>	2	李四	25
<input checked="" type="checkbox"/>	3	王五	27
<input checked="" type="checkbox"/>	4	赵六	29
<input checked="" type="checkbox"/>	5	田七	30
<input checked="" type="checkbox"/>	6	汾九	20

### 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：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.对编号前面的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：

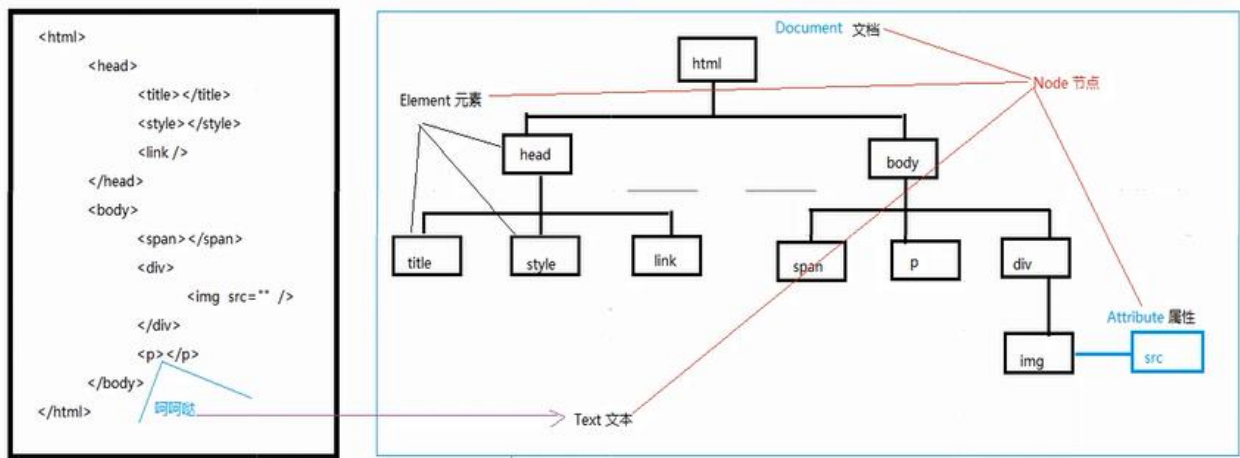
```
<th><input type="checkbox" onclick="checkAll()" id="checkAll"/></th>
```

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 文件

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。

方法	描述
<a href="#">close()</a>	关闭用 <code>document.open()</code> 方法打开的输出流，并显示选定的数据。
<a href="#">getElementById()</a>	返回对拥有指定 <code>id</code> 的第一个对象的引用。
<a href="#">getElementsByName()</a>	返回带有指定名称的对象集合。
<a href="#">getElementsByTagName()</a>	返回带有指定标签名的对象集合。
<a href="#">open()</a>	打开一个流，以收集来自任何 <code>document.write()</code> 或 <code>document.writeln()</code> 方法的输出。
<a href="#">write()</a>	向文档写 HTML 表达式 或 JavaScript 代码。
<a href="#">writeln()</a>	等同于 <code>write()</code> 方法，不同的是在每个表达式之后写一个换行符。

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() 向元素添加新的子节点，作为最后一个子节点。

element.firstChild 返回元素的首个子节点。

element.getAttribute() 返回元素节点的指定属性值。

element.innerHTML 设置或返回元素的内容

element.insertBefore() 在指定的已有的子节点之前插入新节点。

element.lastChild 返回元素的最后一个子元素。

element.setAttribute() 把指定属性设置或更改为指定值。

## 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

属性 / 方法	描述
<a href="#">attr.isId</a>	如果属性是 id 类型，则返回 true，否则返回 false。
<a href="#">attr.name</a>	返回属性的名称。
<a href="#">attr.value</a>	设置或返回属性的值。
<a href="#">attr.specified</a>	如果已指定属性，则返回 true，否则返回 false。
<a href="#">nodemap.getNamedItem()</a>	从 NamedNodeMap 返回指定的属性节点。
<a href="#">nodemap.item()</a>	返回 NamedNodeMap 中位于指定下标的节点。
<a href="#">nodemap.length</a>	返回 NamedNodeMap 中的节点数。
<a href="#">nodemap.removeNamedItem()</a>	移除指定的属性节点。
<a href="#">nodemap.setNamedItem()</a>	设置指定的属性节点（通过名称）。

## 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，显示效果如下：

表中动态加载该省份下所有的城市，显示效果如下：

会员注册			
用户名			
密码			
确认密码			
email			
姓名			
籍贯	湖南 ▼	长沙 ▼	
性别	<input type="radio"/> 男 <input type="radio"/> 女	长沙	
出生日期		株洲	
验证码		郴州	
		岳阳	
<input type="button" value="注册"/>			

## 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：

```
<!--2.确定事件，通过函数传参的方式拿到改变后的城市-->  
  
    <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

**JavaScript 对象**  
JS Array  
JS Boolean  
JS Date  
JS Math  
JS Number  
JS String  
JS RegExp

#### Array 对象

数组的创建：

## Array 对象

Array 对象用于在单个的变量中存储多个值。

创建 Array 对象的语法：

```
new Array();  
new Array(size);  
new Array(element0, element1, ..., elementn);
```

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：

创建 Boolean 对象的语法：

```
new Boolean(value);           //构造函数  
Boolean(value);               //转换函数
```

如果 value 不写，那么默认创建的结果为 false

## Date 对象

getTime() 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()          找到一个或多个正则表达式的匹配。

substr()          从起始索引提取字符串中指定数目的字符。

substring() 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 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键的 JavaScript 对象

函数	描述
<a href="#">decodeURI()</a>	解码某个编码的 URI。
<a href="#">decodeURIComponent()</a>	解码一个编码的 URI 组件。
<a href="#">encodeURI()</a>	把字符串编码为 URI。
<a href="#">encodeURIComponent()</a>	把字符串编码为 URI 组件。
<a href="#">escape()</a>	对字符串进行编码。
<a href="#">eval()</a>	计算 JavaScript 字符串，并把它作为脚本代码来执行。
<a href="#">getClass()</a>	返回一个 JavaScript 对象的 JavaClass。
<a href="#">isFinite()</a>	检查某个值是否为有穷大的数。
<a href="#">isNaN()</a>	检查某个值是否是数字。
<a href="#">Number()</a>	把对象的值转换为数字。
<a href="#">parseFloat()</a>	解析一个字符串并返回一个浮点数。
<a href="#">parseInt()</a>	解析一个字符串并返回一个整数。
<a href="#">String()</a>	把对象的值转换为字符串。
<a href="#">unescape()</a>	对由 <a href="#">escape()</a> 编码的字符串进行解码。