

JavaScript入门

今日内容介绍

- ◆ 使用 JS 完成简单的数据校验
- ◆ 使用 JS 完成图片轮播效果
- ◆ 使用 JS 完成页面定时弹出广告
- ◆ 使用 JS 完成表单校验

今日内容学习目标

- ◆ 掌握 JavaScript 的基本语法
- ◆ 掌握 JavaScript 的对象获取
- ◆ 掌握 JavaScript 标签的基本操作
- ◆ 使用 JS 可以获得指定元素
- ◆ 使用 JS 可以对元素的标签体进行操作
- ◆ 使用 JS 可以对指定元素的样式进行操作（获得或修改）
- ◆ 使用 JS 可以编写各种事件
- ◆ 使用 JS 可以编写定时程序

第1章 案例：使用 JS 完成注册页面的校验

1.1 案例介绍

用户在提交表单时，需要对用户的填写的数据进行校验。本案例只对用户名、密码、确认密码和邮

箱进行校验。

1.2 相关知识点：

1.2.1 JavaScript 的概述

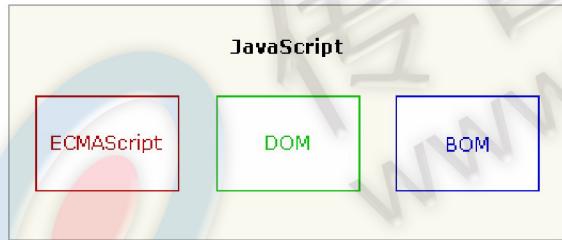
1.2.1.1 什么是 JavaScript

JavaScript 是 web 上一种功能强大的编程语言，用于开发交互式的 web 页面。它不需要进行编译，而是直接嵌入在 HTML 页面中，由浏览器执行。

- 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）。

JavaScript 的组成：

- 核心 (ECMAScript)
- 文档对象模型 (DOM)
- 浏览器对象模型 (BOM)



- ECMAScript: 语法，语句.
- BOM: 浏览器对象
- DOM: Document Object Model. 操作文档中的元素和内容.

1.2.1.2 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、对浏览器事件做出响应、读写 HTML 元素、验证提交数据、检测访客的浏览器信息等。

1.2.1.3 JavaScript 的引入

在 HTML 文件中引入 JavaScript 有两种方式，一种是在 HTML 文档直接嵌入 JavaScript 脚本，称为内嵌式，另一种是链接外部 JavaScript 脚本文件，称为外联式。对他们的具体讲解如下：

- 1) 内嵌式，在 HTML 文档中，通过<script>标签引入，如下

```
<script type="text/javascript">  
    //此处为 JavaScript 代码  
</script>
```

2) 外联式，在 HTML 文档中，通过<script src="">标签引入.js 文件，如下：

```
<script src="1.js" type="text/javascript" charset="utf-8"></script>
```

1.2.2 基本语法

1.2.2.1 变量

1) 在使用 JavaScript 时，需要遵循以下命名规范：

- 必须以字母或下划线开头，中间可以是数字、字符或下划线
- 变量名不能包含空格等符号
- 不能使用 JavaScript 关键字作为变量名，如：function
- JavaScript 严格区分大小写。

2) 变量的声明

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

3) 变量的赋值

```
var 变量名 = 值; //JavaScript 变量是弱类型，及同一个变量可以存放不同类型的数据
```

1.2.2.2 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 ,只有一个专用值 null，表示空，一个占位符。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。
- alert(null == undefined); //输出 "true"，尽管这两个值相等，但它们的含义不同。
 - Boolean，有两个值 true 和 false
 - Number，表示任意数字
 - String，字符串由双引号 ("") 或单引号 ('') 声明的。JavaScript 没有字符类型

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。
- JavaScript 是基于对象而不是面向对象。对象类型的默认值是 null.
- JavaScript 提供众多预定义引用类型（内置对象）。

1.2.2.3 运算符

JavaScript 运算符与 Java 运算符基本一致。

- 算术运算符

运算符	描述	例子	结果
+	加	x=y+2	x=7
-	减	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求余数 (保留整数)	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=-y	x=4

- 赋值运算符

运算符	例子	等价于	结果
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

- 比较运算符

运算符	描述	例子
==	等于	x==8 为 false
===	全等 (值和类型)	x === 5 为 true; x === "5" 为 false
!=	不等于	x != 8 为 true
>	大于	x > 8 为 false
<	小于	x < 8 为 true
>=	大于或等于	x >= 8 为 false
<=	小于或等于	x <= 8 为 true

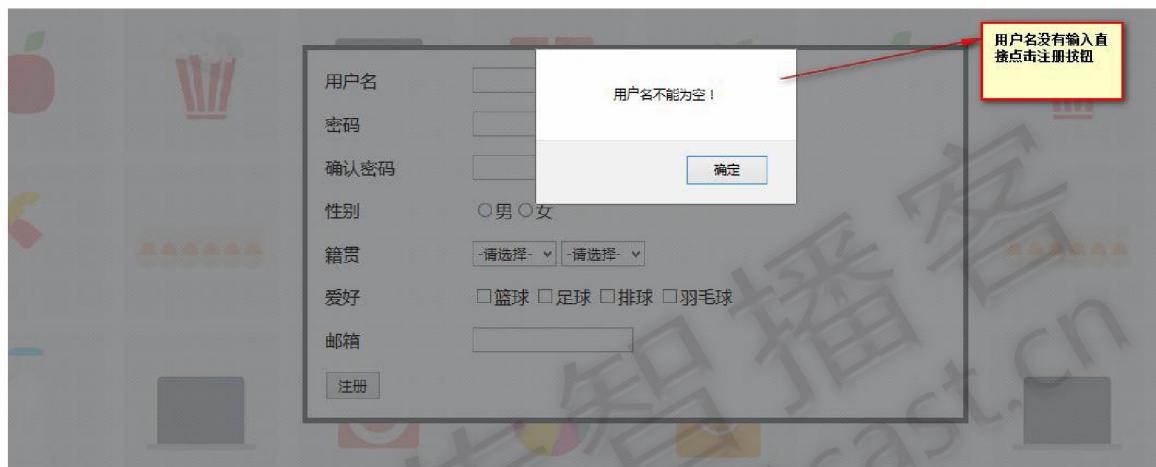
- 逻辑运算符

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
	or	(x == 5 y == 5) 为 false
!	not	!(x == y) 为 true

1.2.2.4 基本操作

- `alert()` : 向页面中弹出一个提示框!!
- `innerHTML` : 向页面的某个元素中写一段内容，将原有的东西覆盖
- `document.write()` : 向页面中写内容

1.3 案例分析



1.4 案例实现

- 步骤 1: 表单`<form>` 添加提交事件

```
65<div class="regist">
66<form action="../index.html" onsubmit="return check();">
67<table width="600" height="350px">
```

- 步骤 2: 编写 `check()` 函数，进行校验

```
<script type="text/javascript">
    function check() {
        //1 用户名
        var loginName = document.getElementById("loginnameId").value;
        if(loginName == "") {
            alert("用户名不能为空");
            return false;
        }
        //2 密码
        var loginpwd = document.getElementById("loginpwdId").value;
        if(loginpwd == "") {
            alert("密码不能为空");
            return false;
        }
    }
</script>
```

```
//3 确认密码
var reloginpwd = document.getElementById("reloginpwdId").value;
if(reloginpwd != loginpwd) {
    alert("密码和确认密码不一致");
    return false;
}

//4 邮箱
var email = document.getElementById("emailId").value;
if(! /^[0-9a-zA-Z_-]+@[0-9a-zA-Z_-]+\.\w{2,}$/.test(email)) {
    alert("邮箱格式不正确");
    document.getElementById("emailId").focus(); //提示信息之后获得焦点
    return false;
}

return true;
}
</script>
```

1.5 案例总结

1.5.1 JS 中正则匹配的方式：

正则的匹配：

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。

1.5.2 JS 中的函数编写方式：

函数：实现一定功能的代码块，类似与 Java 中的方法。关键字 function，函数名自定义。

```
<script type="text/javascript">
    //方式 1： 声明函数
    function demo01() {
        alert("案例 1");
    }
    // 方式 1： 调用函数
    demo01();

    //方式 2： 声明匿名函数
```

```
var demo02 = function(){
    alert("案例 2");
};

//方式 2：调用函数
demo02();

</script>
```

第2章 案例：轮播图

2.1 案例需求

第一天完成首页中，“轮播图”只提供一张图片进行显示。现需要编写程序，完成自动切换图片功能。



2.2 相关知识点：定时器 **setInterval**

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

参数 1: `code` 必需。执行的函数名或执行的代码字符串。

参数 2: `millisec` 必须。时间间隔，单位：毫秒。

返回值：一个可以传递给 `window.clearInterval()` 从而取消对 `code` 的周期性执行的值。

例如：

* 方式 1：函数名， `setInterval(show, 100);`

* 方式 2：函数字符串， `setInterval("show()", 100);`

- `window` 对象提供都是全局函数，调用函数时 `window` 可以省略。

- `window.setInterval()` 等效 `setInterval()`

2.3 案例分析

1. 编写 html 页面，为页面设置加载事件 `onload`
2. 编写事件触发函数
3. 获得轮播图图片
4. 开启定时器，2000 毫秒重新设置图片的 `src` 属性

2.4 案例实现

- 步骤 1：为轮播图 `img` 标签添加 `id` 属性

```
101     <div style="width: 100%;">  
102           
103     </div>
```

- 步骤 2：编写 js 代码，页面加载触发指定函数

```
<script type="text/javascript">  
    // 加载成功启动监听器，5 秒执行一次  
    window.onload = function () {  
        setInterval(changeImage, 5000);  
    }  
    //在 3 张照片之间切换  
    var num = 1;  
    function changeImage() {  
        if (num >= 3) {  
            num = 1;  
        }  
        var imageObj = document.getElementById("imgId");  
        imageObj.src = "../img/" + ++num + ".jpg";  
    }  
</script>
```

第3章 案例：定时弹广告

3.1 案例需求

在平日浏览网页时，页面一打开 5 秒后显示广告，然后 5 秒后再隐藏广告。



3.2 相关知识点

3.2.1 JavaScript 定时器: setTimeout

- `setTimeout()` 在指定的毫秒数后调用函数或执行代码片段。

```
setTimeout(code, millisec)
    code 必需。要调用的函数或要执行的代码字符串。
    millisec 必需。在执行代码前需等待的毫秒数。
```
- `setInterval()` 以指定周期执行函数或代码片段。(上一个案例已经讲解)
- `clearInterval()` 取消由 `setInterval()` 设置的 `timeout`。
- `clearTimeout()` 取消由 `setTimeout()` 方法设置的 `timeout`。(本案例不使用，此处一并讲解)

<u>setInterval()</u>	按照指定的周期(以毫秒计)来调用函数或计算表达式。
<u>setTimeout()</u>	在指定的毫秒数后调用函数或计算表达式。
<u>clearInterval()</u>	取消由 <code>setInterval()</code> 设置的 <code>timeout</code> 。
<u>clearTimeout()</u>	取消由 <code>setTimeout()</code> 方法设置的 <code>timeout</code> 。

3.2.2 JavaScript 样式获得或修改

- 获得或设置样式

```
obj.style.属性      , 获得指定“属性”的值。
```

obj.style.属性=值，给指定“属性”设置内容。

如果属性由多个单词使用“-”连接，需要将“-”删除，并将后一个单词首字母大写。

例如：background-color 需要改成 backgroundColor

● 实例：

```
<div id="divId" style="height:100px;width:100px;margin:10px;padding:20px"></div>

<script type="text/javascript">
    //1 获得 div 对象
    var divObj = document.getElementById("divId");
    //2 获得高度
    // * divObj.style.height 数据为"100px"
    // * 使用 parseInt()，将字符串“100px”转换成数字“100”
    var height = window.parseInt(divObj.style.height);
    //3 将原始高度翻倍，再设置给 div。
    // * 注意：必须添加单位，否则无效
    divObj.style.height = height * 2 + "px";
</script>
```

3.3 案例分析

1. 页面加载成功后触发 `onload()` 事件
2. 加载成功后，触发延迟定时器，5 秒后，开始显示广告。
3. 显示广告开始后，5 秒后再次隐藏广告

3.4 案例实现

● 步骤 1：在页面中，添加广告位 `div`，并设置页面加载事件

```
<body onload="init()">
    <!-- 整体的 DIV -->
    <div>
        <!-- 定时弹出广告的 div -->
        <div id="divAd" style="width:99%;display: none;">
            
        </div> ...
    </div>
```

● 步骤 2：编程 JS 实现

```
<!-- 在 html 页面中引入 js 文件-->
<script src="ad.js"></script>
<!--ad.js 内容如下-->
```

```

var time;

function init() {
    // 设置定时操作:
    time = setInterval("showAd()", 5000);
}

function showAd() {
    // 获得 div 元素
    var divAd = document.getElementById("divAd");
    divAd.style.display = "block";
    // 清除之前的定时操作:
    clearInterval(time);
    // 重新设置一个定时: 5 秒钟隐藏:
    time = setInterval("hideAd()", 5000);
}

function hideAd() {
    // 获得 div 元素
    var divAd = document.getElementById("divAd");
    divAd.style.display = "none";
    clearInterval(time);
}

```

3.5 总结：BOM（Browser Object Model）

Browser 对象
DOM Window
DOM Navigator
DOM Screen
DOM History
DOM Location

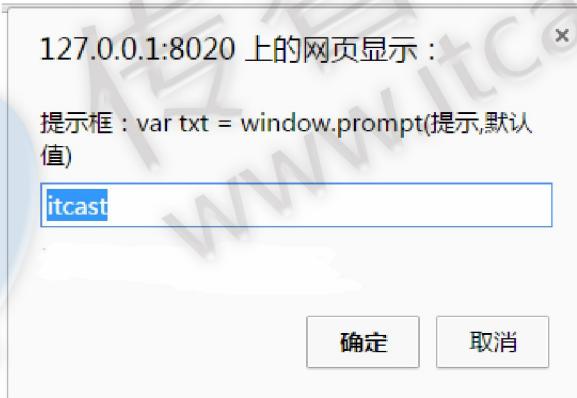
3.5.1 BOM : Window 对象 (掌握)

- 方法：定时器

函数名	描述
setInterval()	按照指定的周期（以毫秒计）来调用函数或计算表达式
clearInterval()	取消由 setInterval() 设置的 timeout。
setTimeout()	在指定的毫秒数后调用函数或计算表达式。
clearTimeout()	取消由 setTimeout() 方法设置的 timeout

- 方法：消息框

函数名	描述
-----	----

<code>alert()</code>	显示带有一段消息和一个确认按钮的警告框。 
<code>confirm()</code>	显示带有一段消息以及确认按钮和取消按钮的确认框。  确认框： 确定返回 true 取消返回 false
<code>prompt()</code>	显示可提示用户输入的提示框。  点击确定获得用户输入数据

● Window 尺寸

1) IE9 (含, 及以上), Chrome、Firefox 等其他浏览器获得方式

`window.innerHeight` - 浏览器窗口的内部高度

`window.innerWidth` - 浏览器窗口的内部宽度

2) Internet Explorer 8、7、6、5

`document.documentElement.clientHeight`

`document.documentElement.clientWidth`

或者

`document.body.clientHeight`

`document.body.clientWidth`

兼容所有浏览器的 JS 实现方案：

```
var w=window.innerWidth
    || document.documentElement.clientWidth
    || document.body.clientWidth;

var h=window.innerHeight
    || document.documentElement.clientHeight
    || document.body.clientHeight;
```

3.5.2 BOM : Location 对象

- `href` 属性：设置或返回完整的 URL。

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

```
<script type="text/javascript">
    function change() {
        location.href = "http://www.itheima.com";
    }
</script>

<input type="button" value="点我" onclick="change()"/>
```

3.5.3 BOM : History 对象 (了解)

- `go()` 方法：跳转到指定页面
 - `go(-1)` 加载前一个连接，等效 `back()`
 - `go(1)` 加载后一个链接，等效 `forward()`

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

第4章 案例：完善注册表单校验

4.1 案例介绍

昨天我们已经完成了表单数据校验，整个实现过程存在两处不足的地方：

1. 使用弹出框进行提示，用户体验不友好，可以将错误提示信息现在在对应的表单元素后面
2. 在编写程序时存在多处重复代码，为了达到代码的重复利用，将进行内容抽取成，编写自定义函数。

会员注册 USER REGISTER

用户名 用户名不能为空

密码 密码不能为空

确认密码

Email 邮箱格式不正确

姓名

性别 男 女

出生日期

验证码 Z C K x

注册

4.2 相关知识点

4.2.1 标签体内容：innerHTML

- innerHTML - HTML 元素的内部文本

获得: `document.getElementById("divId").innerHTML`

设置: `document.getElementById("divId").innerHTML = "...."`

4.2.2 相关事件

- 常见事件

事件名	描述
onsubmit	提交按钮被点击
onblur	元素失去焦点
onfocus	元素获得焦点

4.3 案例分析

1. 校验不通过，在当前标签后面，红色字体提示
2. 对所有需要校验的表单项进行全部校验
3. 第一个校验不通过的元素获得焦点

编写步骤：

- 1.添加错误提示显示区域 `` (`msg == message` 消息)
- 2.表单元素 `id` 属性
- 3.校验不同，给 `span` 显示错误信息
- 4.第一个不通过的获得焦点

4.4 案例实现

4.4.1 修改 html，添加错误显示区域

```

<form action="..../index.html" onsubmit="return check();">
    <table width="650" height="350px">
        <tr>
            <td colspan="4">
                <font color="#3164af">会员注册</font> USER REGISTER
            </td>
        </tr>
        <tr>
            <td align="right">用户名</td>
            <td colspan="2"><input id="loginnameId" type="text" name="loginname" size="60"/> </td>
            <td><font id="loginnameIdMsg" color="red"></font></td>
        </tr>
        <tr>
    
```

```

        <td align="right">密码</td>
        <td colspan="2"><input id="loginpwdId" type="password" name="loginpwd" size="60"/> </td>
        <td><font id="loginpwdIdMsg" color="red"></font></td>
    </tr>
    <tr>
        <td align="right">确认密码</td>
        <td colspan="2"><input id="reloginpwdId" type="password" name="reloginpwd" size="60"/> </td>
        <td><font id="reloginpwdIdMsg" color="red"></font></td>
    </tr>
    <tr>
        <td align="right">Email</td>
        <td colspan="2"><input id="emailId" type="text" name="email" size="60"/> </td>
        <td><font id="emailIdMsg" color="red"></font></td>
    </tr>
    <tr>
        <td align="right">姓名</td>
        <td colspan="2"><input name="text" name="username" size="60"/> </td>
        <td>&nbsp;</td> <!-- &nbsp; 转义符号，表示一个空格，固定写法-->
    </tr>
    .....
</table>
</form>

```

4.4.2 获得指定的 id 的值

- 提供函数 val(objId)获得指定 id 属性对应元素 value 的值

```

/**
 * 获得指定元素 value 属性的值
 * 例如: var loginName = val("loginnameId");
 * @param objId
 */
function val(objId) {
    return document.getElementById(objId).value;
}

```

4.4.3 错误提示

- 当标签没有校验通过时，给出提示信息。如果校验通过隐藏错误提示信息

```

    /**
     * #2.1 显示错误提示信息
     * 例如: showTip("loginnameIdMsg", "用户名不能为空");
     * @param errObjId
     * @param text
     */
    function showTip(errObjId, text) {
        // 获得错误提示对象，并设置和显示
        var showObj = document.getElementById(errObjId);
        showObj.innerHTML = text;
        showObj.style.display = "block";
    }
    /**
     * #2.2 隐藏错误提示信息
     */
    function hideTip() {
        var showObj = document.getElementById(errObjId);
        showObj.innerHTML = "";
        showObj.style.display = "none";
    }

```

4.4.4 获得焦点

```

    /**
     * #3 获得焦点
     * @param objId
     */
    function focus(objId) {
        // 指定标签获得焦点
        document.getElementById(objId).focus();
    }

```

4.4.5 校验

- 不同的表单元素采用不同的校验规则
 - 如果校验不通过，记录当前表示 id 的属性值
 - 如果校验通过，清除错误提示信息
- 当所有的校验项都校验结束后，统一确定是否都校验成功。如果不成功，第一个不通过获得焦点
 - `focusObjId = focusObjId || "reloginpwdId";` 表示如果之前有记录采用之前的，如果没有采用后面的。

```
var focusObjId; → undefined
```

undefined 转换成 boolean	→ false
false obj	→ obj
obj1 obj2	→ obj1

```

function check() {
    var focusObjId; //记录需要获得焦点标签 id 值
    //1 用户名
    var loginName = val("loginnameId");
    if(loginName == "") {
        showTip("loginnameIdMsg", "用户名不能为空");
        focusObjId = "loginnameId";
    } else {
        hideTip("loginnameIdMsg");
    }

    //2 密码
    var loginpwd = val("loginpwdId");
    if(loginpwd == "") {
        showTip("loginpwdIdMsg", "密码不能为空");
        focusObjId = focusObjId || "loginpwdId"; // ||
    } else {
        hideTip("loginpwdIdMsg");
    }

    //3 确认密码
    var reloginpwd = val("reloginpwdId");
    if(reloginpwd != loginpwd) {
        showTip("reloginpwdIdMsg", "密码和确认密码不一致");
        focusObjId = focusObjId || "reloginpwdId";
    } else {
        hideTip("reloginpwdIdMsg");
    }

    //4 邮箱
    var email = val("emailId");
    if(! /^[0-9a-zA-Z_-]+@[0-9a-zA-Z_-]+\.\w{2,}$/.test(email)) {
        showTip("emailIdMsg", "邮箱格式不正确");
        focusObjId = focusObjId || "emailId";
    } else {
        hideTip("emailIdMsg");
    }

    //5 如果有需要获得焦点，表示校验没有通过
}

```

```

if(focusObjId) {
    focus(focusObjId);
    return false;
}
return true;
}

```

4.5 总结：事件

- 常见事件

事件名	描述
onload	某个页面或图像被完成加载
onsubmit	提交按钮被点击
onclick	鼠标点击某个对象
ondblclick	鼠标双击某个对象
onblur	元素失去焦点
onfocus	元素获得焦点
onchange	用户改变域的内容
onkeydown	某个键盘的键被按下
onkeypress	某个键盘的键被按下或按住
onkeyup	某个键盘的键被松开
onmousedown	某个鼠标按键被按下
onmouseup	某个鼠标按键被松开
onmouseover	鼠标被移到某元素之上
onmouseout	鼠标从某元素移开
onmousemove	鼠标被移动

- 参考：

HTML 代码

```

<input id="e01" type="text" /><span id="textMsg"></span> <br/>
<hr/>
<div id="e02" ></div><span id="divMsg"></span> <br/>
<hr/>
<input id="e03" type="button" value="可以点击"/><span id="buttonMsg"></span>
<br/>

```

JavaScript 代码

```

<script type="text/javascript">

// 页面加载事件：当整个 html 页面加载成功调用
window.onload = function() {
    // 文本框事件
    var e01 = document.getElementById("e01");

```

```
e01.onfocus = function(){
    html("textMsg", "文本框获得焦点: focus");
}

e01.onblur = function(){
    html("textMsg", "文本框失去焦点: blur");
}

e01.onkeydown = function(){
    html("textMsg", "键盘按下: keydown");
}

e01.onkeypress = function(){
    append("textMsg", "键盘按: keypress");
}

e01.onkeyup = function(){
    append("textMsg", "键盘弹起: keyup");
}

// 鼠标事件
var e02 = document.getElementById("e02");
var i = 0;

e02.onmouseover = function(){
    html("divMsg", "鼠标移上: mouseover");
}

e02.onmousemove = function(){
    //html("divMsg", "鼠标移动: mousemove , " + i++);
}

e02.onmouseout = function(){
    html("divMsg", "鼠标移出: mouseout");
}

e02.onmousedown = function(){
    html("divMsg", "鼠标按下: mousedown");
}

e02.onmouseup = function(){
    html("divMsg", "鼠标弹起: mouseup");
}

// 按钮事件
var e03 = document.getElementById("e03");
e03.onclick = function () {
    html("buttonMsg", "单击: click");
}

e03.ondblclick= function () {
    html("buttonMsg", "双击: dblclick");
}
```

```

};

/**
 * 指定位置显示指定信息
 * @param objId , 元素的 id 属性值
 * @param text, 需要显示文本信息
 */
function html(objId,text){
    document.getElementById(objId).innerHTML = text;
}

/**
 * 指定位置追加指定信息
 * @param objId , 元素的 id 属性值
 * @param text, 需要显示文本信息
 */
function append(objId,text){
    document.getElementById(objId).innerHTML += text;
}

</script>

```

● event 属性

属性名	描述
clientX	返回当事件被触发时，鼠标指针的水平坐标。
clientY	返回当事件被触发时，鼠标指针的垂直坐标。
keyCode	返回当事件被触发时，键盘输入 ASCII 码

● event 方法:

方法名	描述
preventDefault()	阻止浏览器默认行为
stopPropagation()	阻止事件的传播

● 阻止浏览器默认行为

```

<a href="http://www.itheima.com" onclick="one()">黑马程序员</a><br/>
<a href="http://www.itcast.cn" onclick="return two()">传智播客</a><br/>
<script type="text/javascript">

    function one() {
        alert("我之后， 黑马官网继续访问");
    }

    function two(event) {
        alert("我之后， 传智官网不再访问");
        //方式 1:
    }

```

```
//return false;  
//方式 2:  
var event = event || window.event;  
event.preventDefault();  
}  
</script>
```

● 阻止事件的传播

```
<!--区域 1-->  
<div id="e1" style="width:200px ; height: 200px; background-color: #f00;">  
    <div id="e2" style="width:100px ; height: 100px; background-color: #0f0;"></div>  
</div>  


---

  
<!--区域 2-->  
<div id="e3" style="width:200px ; height: 200px; background-color: #f00;">  
    <div id="e4" style="width:100px ; height: 100px; background-color: #0f0;"></div>  
</div>  
  
<script type="text/javascript">  
    var e1 = document.getElementById("e1");  
    var e2 = document.getElementById("e2");  
    var e3 = document.getElementById("e3");  
    var e4 = document.getElementById("e4");  
  
    //设置“区域 1”事件  
    e1.onclick = function(){  
        alert("e1");  
    }  
    e2.onclick = function(){  
        alert("e2, 同时 e1 也触发");  
    }  
  
    //设置“区域 2”事件  
    e3.onclick = function(){  
        alert("e3");  
    }  
    e4.onclick = function(event){  
        alert("e4, e3 不触发");  
        var event = event || window.event;  
        event.stopPropagation();  
    }  
</script>
```