

# EasyCheck 验证插件使用手册

## 目录

1、引入 EasyCheck.css , 在 jQuery 后引入 EasyCheck.js.....	1
2、在页面使用 EasyCheck 的验证器进行验证.....	3
3、提交表单时验证 .....	4
4、防止客户端表单重复提交功能 .....	4
5、验证通过文本框样式 .....	5
6、禁用验证失败时文本框样式 .....	5
6、自定义消息 .....	5
7、自定义消息内容 .....	6
8、手动清除所有错误提示消息 .....	7
9、高级选项：自定义新验证器——在页面添加一个新的验证器 .....	7
10、高级扩展：扩展 EasyCheck 验证框架，加入新验证器插件 .....	8
11、高级扩展：验证框架提示消息维护管理 .....	9

## 快速开发步骤：

### 1、引入 EasyCheck.css , 在 jQuery 后引入 EasyCheck.js

```
<link rel="stylesheet" type="text/css" href="css/EasyCheck.css"/>

<script type="text/javascript" src="js/jquery.js"> </script>

<script type="text/javascript" src="js/EasyCheck.min.js"> </script>

<script type="text/javascript">

    //可选参数

    EasyCheck.blurChk=true;    //开启失去焦点时验证，false 禁用，默认为 true

    EasyCheck.keyupChk=true;  //开启键盘弹起时验证，false 禁用，默认为 true

    EasyCheck.loadChk=true;   //页面加载完后是否立即开启验证规则( 否则仅在提交表单时验证 ,如果设置为 false ,blurChk 和 keyupChk
无效 ), 默认为 true

</script>
```

### EasyCheck.css 说明：

EasyCheck.css 中定义了：

- 1、验证未通过消息类样式 `.easyccheck_errorInfo`
- 2、验证通过消息类样式 `.easyccheck_okInfo`

3、验证未通过文本框类样式 `.easycheck_errorInput`

4、验证通过文本框类样式 `.easycheck_okInput`

四种状态下消息和文本框的 CSS 样式。

根据实际项目需要可进行调整，**请将**`.easycheck_okInput` 类样式修改为您文本框默认样式；  
**将**`.easycheck_errorInput` 类样式修改为错误提示您需要的样式。或者，在页面重新定义该类样式，  
**以替换默认样式。**

```
/* CSS Document */
/*验证未通过的div样式*/
.easycheck_errorInfo {
    padding: 2px 8px;
    padding-left: 8px;
    height: 30px;
    font-family: 微软雅黑, Arial, Helvetica, sans-serif;
    margin-left: 10px;
    font-size: 15px;
    line-height: 30px;
    background-color: #FF6600;
    color: #fff;
    display: inline;
    font-weight: bold;
}
/*验证通过的div样式*/
.easycheck_okInfo {
    padding: 3px 8px;
    height: 30px;
    font-family: 微软雅黑, Arial, Helvetica, sans-serif;
    margin-left: 10px;
    font-size: 15px;
    line-height: 30px;
    background-color: #7AC935;
    color: #fff;
    display: inline;
    font-weight: bold;
}
/*验证未通过的文本框样式*/
.easycheck_errorInput {
    height: 30px;
    font-size: 14px;
    text-shadow: 0 0 0 rgba(0, 0, 0, 0.3);
    border: 1px solid #DD080A;
    text-indent: inherit;
    font-weight: normal;
    line-height: 30px;
    background: #fff;
    font-family: "微软雅黑", sans-serif, Verdana, Arial, Helvetica, Tahoma, "宋体";
    margin-right: 10px;
    width: 325px;
    margin: 0px 0 0 10px;
    letter-spacing: 1px;
}
/*验证通过的默认文本框样式,与文本框默认样式保持一致！*/
.easycheck_okInput {
    height: 30px;
    font-size: 14px;
    text-shadow: 0 0 0 rgba(0, 0, 0, 0.3);
    border: 1px solid #cfcfc9;
    text-indent: inherit;
    font-weight: normal;
    line-height: 30px;
    background: #fff;
```

```
font-family: "微软雅黑", sans-serif, Verdana, Arial, Helvetica, Tahoma, "宋体";
margin-right: 10px;
width: 325px;
margin: 0px 0 0 10px;
letter-spacing: 1px;
}
```

## 2、在页面使用 EasyCheck 的验证器进行验证

- **class 类验证器：**

**required** 必填 `<input type="text" name="name" class="required" />`

email 邮箱 `<input type="text" name="name" class="email" />`

```
url      URL      <input type="text" name="name" class=" url" />
```

number 数字 `<input type="text" name="name" class="number" />`

**integer**      整数      `<input type="text" name="name" class="integer" />`

**同时使用多个类验证器，用空格分隔：**

不能为空，并且为邮箱 `<input type="text" name="name" class="required email" />`

- **attribute 属性验证器:**

**equalTo** 值必须和 Id 为 ElementId 指定的元素相等 `<input type="password" name="name" equalTo="ElementId"/>`

**equallength** 值长度必须等于 equallength `<input type="password" name="name" equallength="4"/>`

**maxlength** 最大字符长度不能大于 maxlength `<input type="text" name="name" maxlength="20"/>`

**minlength** 最小字符长度不能少于 minlength `<input type="text" name="name" minlength="6"/>`

max      数字不能大于 max      `<input type="text" name="name" max="20"/>`

**min** 数字不能小于 min `<input type="text" name="name" min="2"/>`

**extension** 验证扩展名，多个扩展名使用英文逗号分隔，默认为"png,jpeg,jpg,gif"

```
<input type="text" name="name" extension="" />
```

reg 自定义正则验证 `<input type="text" name="name" reg="[A-Z]*"/>`

VC 使用 Ajax 请求 vc 指定的 **URL**，进行验证码检测，URL 返回 true 代表通过，false 代表未通过

```
<input type="text" vc="chkvc.jsp" name="vc" />
```

## 服务器端自定义处理 Demo :

```
String vc = request.getParameter("vc"); //通过验证码文本框名称获得输入的数据
String res = "false";
if (vc != null && vc.equals(session.getAttribute("randomNumber"))) {
```

```
        res = "true";
    }

    out.print(res);  输出true代表通过，false代表未通过
%>
```

说明，默认情况下为了避免不必要的服务器请求，验证码验证**只在提交表单时进行**，不在键盘弹起和失去焦点时进行验证的参数。实现代码：

**EasyCheck.easyCheckIgnore["vc"]=true;** //vc 验证码规则，键盘弹起和失去焦点时不验证，只在表单提交时验证

**easyCheckIgnor** 参数可以设置弹起和焦点验证时的忽略验证器，可根据需要修改为 false 或注释掉，代表进行键盘弹起和失去焦点时验证。

同时写上 minlength 与 maxlength，此时将进行范围提示：

```
<input type="password" value="" name="urepwd" size="20" class="txt required" equalto="upwd" maxlength="12" minlength="6"/>
```

同时写上 min 与 max，此时将进行范围提示：

```
<input type="password" value="" name="urepwd" size="20" class="txt required" max="20" max="40"/>
```

### 3、提交表单时验证

提交表单时进行验证，为 form 表单添加指定 **id 属性**和 **onsubmit="return easyCheckForm(this)"**事件即可

```
<form action="login.action" onsubmit="return easyCheckForm(this)" id="regForm">
```

### 4、防止客户端表单重复提交功能

EasyCheck 默认开启了客户端防止重复提交功能。防止在用户验证通过提交数据过程中，由于网络未响应，用户多次点击提交等原因，导致重复提交数据功能。默认用户点击 submit 按钮提交表单过程中将禁用 submit 提交按钮。

如果在特殊场景下需要禁用该功能，可在引入 EasyCheck.js 后，设置 **EasyCheck.easyCheckSubmitDisable** 参数值为 **false** 即可禁用防重复提交功能：

```
EasyCheck.easyCheckSubmitDisable=false;  //取消提交按钮禁用功能
```

### Firefox 下特别说明：

由于 Firefox 浏览器的从缓存加载数据时的记忆原因，如果提交数据后，通过**点击浏览器后退按钮**回到网页，提交按钮将依然显示为禁用状态。

解决方法为给提交按钮加上 **autocomplete="off"** 的属性即可。**autocomplete** 属性作用说明：

此为了屏蔽浏览器表单默认的记忆功能。淘宝，百度的搜索框也有该属性。autocomplete 属性是非标准的，首先在 IE5 中加入，后 其它浏览器 都 支持。html5 也将其列表标准。

除此之外，如果不希望通修改 html 页面为提交按钮加 autocomplete="off"属性来实现此功能，EasyCheck 同样支持通过 JS 代码实现修正 Firefox 的功能：

```
//方法一：直接设置 Firefox 下后退后不禁用的按钮 id 数组，可指定多个
EasyCheck.removeDisableBtn=['submitId'];

//方法二：设置 Firefox 下后退后不禁用的 formId 数组，可指定多个 form 表单的 ID，在表单中的所有 submit
按钮在后退后自动转为正常

EasyCheck.removeDisableForm=['formId'];

//方法三：设置强制将页面所有 form 表单中的 submit 按钮启用，默认值为 false

EasyCheck.removeDisable=true;  //( 该参数会将所有所有 from 下禁用的 submit 按钮启用，所有
如果确定项目页面没有默认需要禁用的 submit 按钮，该设置最为方便，可在 EasyCheck.js 源文件中将
removeDisable 参数配置为 true 达到同样效果 )

    以上方法支持混合使用。
```

## 5、验证通过文本框样式

默认情况下验证通过，文本框需要恢复到正常的样式，EasyCheck 会使用 EasyCheck.css 中的 *easycheck\_okInput* 样式来重新定义文本框样式。所以需要将 *easycheck\_okInput* 样式同默认样式设置为一致。

在 JS 中还可以通过 EasyCheck.okcss 属性直接将已有的默认样式指定为全局的验证通过文本框样式，替代 easycheck\_okInput。

```
EasyCheck.okcss="txt";
```

## 6、禁用验证失败时文本框样式

默认情况下验证失败时，除了显示错误提示信息外，文本框会使用.easycheck\_errorInput 类样式（验证未通过文本框样式）显示，如果需要禁用该显示效果，可以通过给文本框对象加入 ecss = "no" 属性实现禁用（EasyCheck CSS）。

```
<textarea name=" content" class="required" style="width: 400px;height:
100px;border: 1px solid #D4D0C8;" ecss="no"></textarea>
```

在页面元素过多时，还可通过全局参数 EasyCheck.ecss 设置禁用页面所有验证对象的验证未通过样式：

```
EasyCheck.ecss="no";
```

## 6、自定义消息

EasyCheck 支持手动指定默认提示信息，错误提示信息，正确提示信息。

默认情况下，EasyCheck 会在验证失败的文本框的后面自动显示提示消息，如果希望将提示信息显

示在指定的位置，则可设置如下：

- 在页面合适位置创建消息提示 div，为显示消息的 div 指定 id，命名格式必须为 `error_ElementName ( error_验证元素的 name )`
- [info 属性为可选属性，如果设置了，会作为提示信息的前缀。]

EasyCheck 验证插件的提示消息会显示在您 div 指定的位置。

如：

```
<tr>

    <td align="left"  width="300px">

        <label class="lbl"><div style="color:#FF0000; display:inline">*</div>登录邮箱</label>

        <div id="error_uemail" info="登录邮箱"></div>

    /td>

</tr>

<tr>

    <td align="left"><input  type="text"  name="uemail" value="" class="txt required_email" size="20"  /> </td>

</tr>
```

如果需要指定默认提示信息和正确提示信息，则提供如下 id 命名的 DIV 即可：

- 默认提示：`default_验证元素的 name`
- 正确提示：`ok_验证元素的 name`

```
<div id="default_uname2" style="display: inline;">必填，字母开头，只能包含字母和数字</div>

<div id="ok_uname2" style="display: inline;">正确</div>
```

正确提示会使用 `easycheck_okInfo` 样式。

## 7、自定义消息内容

在页面引入 EasyCheck.js 后，使用如下语法，可修改指定验证规则的提示消息内容：

```
EasyCheck.msg['required']=" is required";
```

可根据需要修改，EasyCheck. msg 列表中默认的消息名称和默认值如下：

- `required` 不能为空
- `email` 邮箱格式不正确
- `url` 网址有误
- `number` 必须为数字
- `digits` 必须为整数
- `equalto` 输入不一致
- `equallength` 长度必须为{0}位

- maxlength 长度不能小于{0}
- minlength 长度不能大于{0}
- maxlength minlength 长度必须在{0}到{1}之间
- max 不能小于
- min 不能大于
- max min 值必须在{0}到{1}之间
- regExp 格式有误
- extension 文件扩展名只能为{0}
- vc 输入有误

## 8、手动清除所有错误提示信息

例如验证表单在弹出层中时，打开层时，清空层中表单之前的提示信息。

直接调用

EasyCheck.clearAllError();

即可。

## 9、高级选项：自定义新验证器——在页面添加一个新的验证器

假设页面需要一个检测进行用户名是否存在的类验证器，则可直接定义

```
//2、定义验证函数(o,e)，检测用户名是否存在
function checkExists(o,e){
    /*
    自定义新验证插件函数，调用EasyCheck.addChkMethod(o,e,chkCode,msg)实现向系统注册新验证插件
    o      触发事件的元素
    e      事件对象
    chkCode 实现验证的回调函数，返回true或false作为验证是否通过的结果
    msg    提示信息
    */
    return EasyCheck.addChkMethod(o,e,
        function(o){ //o,验证对象
            var val=$(o).val();
            var res=false; //结果
            dwr.engine.setAsync(false); //禁用DWR异步AJAX
            UserInfoDWR.checkEmail(val,function(d){
                res=d;
            });
            return res;
        },
        "该名称已被使用！");
}
//注册类验证器（验证器名称，验证函数）
EasyCheck.chkList.push(new EasyCheck.ChkRule("exists",checkExists));
```



10、高级扩展：扩展 EasyCheck 验证框架，加入新验证器插件

轻松两步修改 EasyCheck 验证框架，加入自定义新验证器插件！

1、自定义新的验证函数

```
/*
    自定义新验证插件函数，调用 EasyCheck.addChkMethod(o,e,chkCode,msg)实现向系统注册新验证插件
    o 触发事件的元素
    e 事件对象
    chkCode 验证回调函数
    msg 提示消息
    蓝色部分为具体性实现部分
*/
function checkNew(o,e){
    return EasyCheck.addChkMethod(o,e,
        function(o){
            //验证实现，返回 true 或 false：true 代表验证通过；false 代表未通过，将显示 msg 的消息
            // var val=$(o).val();
            //return $.trim(val)!="";
        },
        "验证失败时的消息字符串");
}
```

2、注册自定义验证函数

```
/*
    向系统注册验证对象
    在 EasyCheck 中找到 EasyCheck.chkList 数组，在数组中根据验证函数类别，加入新验证对象
    （ 注册的类或属性名称，注册的新验证插件函数，[属性验证标识 true] ）
*/
/*注册系统类验证器*/
new EasyCheck.ChkRule(" checkNewName ", checkNew )
/*注册系统属性验证器*/
new EasyCheck.ChkRule(" checkNewName ", checkNew ,true),
以上注册代码根据验证函数类别二选一
```



3、如果注册的验证器**只需要在提交表单时验证**（如验证码，无需失去焦点或键盘弹起验证），则可注册如下代码（**蓝色部分为失去焦点或键盘弹起时忽略的验证器名称**）

- **EasyCheck.easyCheckIgnore** 参数类指定忽略验证的规则，设置后同时忽略失去焦点事件和键盘弹起事件的验证（**只对提交表单验证**）。  
EasyCheck.easyCheckIgnore["**vc**"]=true;
- **EasyCheck.easyCheckBlurIgnore** 参数类指定失去焦点事件忽略验证的规则（**只对键盘弹起和表单验证**）。  
EasyCheck.easyCheckBlurIgnore["**vc**"]=true;
- **EasyCheck.easyCheckKeyupIgnore** 参数类指定键盘弹起事件忽略验证的规则（**只对失去焦点和表单验证**）。  
EasyCheck.easyCheckKeyupIgnore["**vc**"]=true;
- **EasyCheck.easyCheckEleIgnore** 键盘弹起和失去焦点时忽略验证的 DOM 元素名称（**只对提交表单有效**）。  
EasyCheck.easyCheckEleIgnore["**uservc**"]=true;

11、高级扩展：验证框架提示消息维护管理

为了方便对提示消息进行管理和国际化。可以将提示消息定义在 **EasyCheck.msg** 列表中：

```
msg:{  
  
    "required":"不能为空",  
  
    "email":"邮箱格式不正确",  
  
    "url":"网址有误",  
  
    "number":"必须为数字",  
  
    "digits":"必须为整数",  
  
    "equalto":"输入不一致",  
  
    "equallength":"长度必须为{0}位",  
  
    "minlength"[maxlength":"长度必须在{0}到{1}之间",  
  
    "minlength":"长度不能小于{0}",  
  
    "maxlength":"长度不能大于{0}",  
  
    "min"[max":"值必须在{0}和{1}之间",  
  
    "min":"不能小于{0}",  
  
    "max":"不能大于{0}",  
  
    "regExp":"格式有误",  
  
    "extension":"文件扩展名只能为{0}",  
  
    "vc":"输入有误",  
  
    "自定义消息名称":"消息内容，可使用{0}，{1}.....占位符"
```

```
}

```

- 在自定义的新验证函数的消息提示部分，使用 `EasyCheck.msg[“自定义消息名称”]` 来获取消息内容

如：

`EasyCheck.msg["required"]`

```
function checkNew(o,e){
    return EasyCheck.addChkMethod(o,e,
        function(o){
            //验证实现，返回 true 或 false：true 代表验证通过；false 代表未通过，将显示 msg 的消息
            //return $.trim(val)!="";
            },EasyCheck.msg["required"]);
}
```

- 如果消息有占位符（如“长度必须在{0}到{1}之间”），则调用 `EasyCheck.formatMsg(“消息内容”，“占位参数值1”，.....)` 为消息赋值进行格式化

如：

```
EasyCheck.formatMsg(EasyCheck.msg["minlength"][maxlength"],$(o).attr("minlength"),$(o).attr("maxlength"))
```