Atitit 表单验证

目录

[1.1.1. 表单数据校验 1](#_Toc24304)

[2. 自定义错误信息 1](#_Toc14670)

[2.1. 要自定义这些消息的外观和文本, 你必须使用 JavaScript; 不能使用 HTML 和 CSS 来改变. 1](#_Toc22707)

[2.2. 使用 JavaScript校验表单控制原生错误信息的界面外观 3](#_Toc31094)

[2.2.1. 约束校验的 API 3](#_Toc15114)

### 表单数据校验

在 Web 中，你可能会遇见各种不同的表单校验：

* **客户端校验**发生在浏览器端，表单数据被提交到服务器之前，这种方式相较于服务器端校验来说，用户体验更好，它能实时的反馈用户的输入校验结果，这种类型的校验可以进一步细分成下面这些方式：
  + **JavaScript** 校验，这是可以完全自定义的实现方式；
  + HTML5 **内置校验**，这不需要 JavaScript ，而且性能更好，但是不能像JavaScript那样可自定义。
* **服务器端校验**则是发生在浏览器提交数据并被服务器端程序接收之后 —— 通常服务器端校验都是发生在将数据写入数据库之前，如果数据没通过校验，则会直接从服务器端返回错误消息，并且告诉浏览器端发生错误的具体位置和原因，服务器端校验不像客户端校验那样有好的用户体验，因为它直到整个表单都提交后才能返回错误信息。但是服务器端校验是你的应用对抗错误/恶意数据的最后防线，在这之后，数据将被持久化至数据库。当今[所有的服务端框架](https://developer.mozilla.org/zh-CN/docs/learn/Server-side/First_steps/Web_frameworks)都提供了数据**校验**与**清洁**功能（让数据更安全）。

# 自定义错误信息

## 要自定义这些消息的外观和文本, 你必须使用 JavaScript; 不能使用 HTML 和 CSS 来改变.

正如我们上面所看到的例子, 每次我们提交无效的表单数据时, 浏览器总会显示错误信息. 但是显示的信息取决于你所使用的浏览器.

这些自动生成的错误有两个缺点:

* 没有标准可以让 CSS 来改变他们的界面外观.
* 这依赖于他们使用的浏览器环境, 意味着你可能在这种语言的页面里得到另一种语言的错误提示.

| **在英文页面上的法语反馈信息版本** | |
| --- | --- |
| **浏览器** | **渲染** |
| Firefox 17 (Windows 7) | IMG_256 |
| Chrome 22 (Windows 7) | IMG_257 |
| Opera 12.10 (Mac OSX) | IMG_258 |

要自定义这些消息的外观和文本, 你必须使用 JavaScript; 不能使用 HTML 和 CSS 来改变.

HTML5 提供 [constraint validation API](http://www.w3.org/TR/html5/forms.html" \l "the-constraint-validation-api" \o "http://www.w3.org/TR/html5/forms.html#the-constraint-validation-api) 来检测和自定义表单元素的状态. 除此之外,他可以改变错误信息的文本. 让我们快速的看一个例子:

<form>

<label for="mail">I would like you to provide me an e-mail</label>

<input type="email" id="mail" name="mail">

<button>Submit</button></form>

在JavaScript 中, 你调用 [setCustomValidity()](https://developer.mozilla.org/en-US/docs/HTML/HTML5/Constraint_validation" \l "Constraint_API's_element.setCustomValidity()" \o "/en-US/docs/HTML/HTML5/Constraint_validation#Constraint_API's_element.setCustomValidity()) 方法:

var email = document.getElementById("mail");

email.addEventListener("input", function (event) {

if (email.validity.typeMismatch) {

email.setCustomValidity("I expect an e-mail, darling!");

} else {

email.setCustomValidity("");

}});

## 使用 JavaScript校验表单控制原生错误信息的界面外观

如果你想控制原生错误信息的界面外观，或者你想处理不支持HTML内置表单校验的浏览器，则必须使用 Javascript。

### 约束校验的 API

越来越多的浏览器支持限制校验API，并且这逐渐变得可靠。这些 API 由成组的方法和属性构成，可在特定的表单元素接口上调用：

* [HTMLButtonElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLButtonElement)
* [HTMLFieldSetElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLFieldSetElement)
* [HTMLInputElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLInputElement)
* [HTMLOutputElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLOutputElement)
* [HTMLSelectElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLSelectElement)
* [HTMLTextAreaElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLTextAreaElement)

### 约束校验的 API 及属性

| **属性** | **描述** |
| --- | --- |
| validationMessage | 一个本地化消息，描述元素不满足校验条件时（如果有的话）的文本信息。如果元素无需校验（willValidate 为 false），或元素的值满足校验条件时，为空字符串。 |
| validity | 一个 [ValidityState](https://developer.mozilla.org/zh-CN/docs/Web/API/ValidityState" \o "DOM接口 ValidityState 代表了一个元素可以处于的与约束验证相关的有效性状态.) 对象，描述元素的验证状态。详见有关可能的验证状态的文章。 |
| validity.customError | 如果元素设置了自定义错误，返回 true ；否则返回false。 |
| validity.patternMismatch | 如果元素的值不匹配所设置的正则表达式，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| validity.rangeOverflow | 如果元素的值高于所设置的最大值，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| validity.rangeUnderflow | 如果元素的值低于所设置的最小值，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| validity.stepMismatch | 如果元素的值不符合 step 属性的规则，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| validity.tooLong | 如果元素的值超过所设置的最大长度，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| validity.typeMismatch | 如果元素的值出现语法错误，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| validity.valid | 如果元素的值不存在校验问题，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:valid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:valid" \o ":valid CSS 伪类表示内容验证正确的<input> 或其他 <form> 元素。这能简单地将校验字段展示为一种能让用户辨别出其输入数据的正确性的样式。) CSS 伪类，否则命中 [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| validity.valueMissing | 如果元素设置了 required 属性且值为空，返回 true，否则返回 false。  当此属性为 true 时，元素将命中  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!) CSS 伪类。 |
| willValidate | 如果元素在表单提交时将被校验，返回 true，否则返回 false。 |

### 约束校验 API 的方法

| **方法** | **描述** |
| --- | --- |
| checkValidity() | 如果元素的值不存在校验问题，返回 true，否则返回 false。如果元素校验失败，此方法会触发invalid 事件。 |
| HTMLFormElement.reportValidity() | 如果元素或它的子元素控件符合校验的限制，返回 true . 当返回为 false 时, 对每个无效元素可撤销 invalid 事件会被唤起并且校验错误会报告给用户 。 |
| setCustomValidity(message) | 为元素添加一个自定义的错误消息；如果设置了自定义错误消息，该元素被认为是无效的，则显示指定的错误。这允许你使用 JavaScript 代码来建立校验失败，而不是用标准约束校验 API 所提供的。这些自定义信息将在向用户报告错误时显示。  如果参数为空，则清空自定义错误。 |

对于旧版浏览器，可以使用 [polyfill（例如 Hyperform](https://hyperform.js.org/" \o "https://hyperform.js.org/)），来弥补其对约束校验 API 支持的不足。既然你已经使用 JavaScript，在您的网站或 Web 应用程序的设计和实现中使用 polyfill 并不是累赘。

#### 使用约束校验 API 的例子

让我们看看如何使用这个 API 来构建自定义错误消息。首先，HTML：

<form novalidate>

<p>

<label for="mail">

<span>Please enter an email address:</span>

<input type="email" id="mail" name="mail">

<span class="error" aria-live="polite"></span>

</label>

</p>

<button>Submit</button></form>

这个简单的表单使用 [novalidate](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/form" \l "attr-novalidate) 属性关闭浏览器的自动校验；这允许我们使用脚本控制表单校验。但是，这并不禁止对约束校验 API的支持或是以下 CSS 伪类：[:valid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:valid" \o ":valid CSS 伪类表示内容验证正确的<input> 或其他 <form> 元素。这能简单地将校验字段展示为一种能让用户辨别出其输入数据的正确性的样式。)、[:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid" \o "此页面仍未被本地化, 期待您的翻译!)、[:in-range](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:in-range" \o "该伪类用于给用户一个可视化的提示，表示输入域的当前值处于允许范围内。) 、[:out-of-range](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:out-of-range" \o "该伪类用于给用户一个可视化的提示，表示输入域的当前值处于允许范围外。) 的应用。这意味着，即使浏览器在发送数据之前没有自动检查表单的有效性，您仍然可以自己做，并相应地设置表单的样式。

[aria-live](https://developer.mozilla.org/en-US/docs/Accessibility/ARIA/ARIA_Live_Regions" \o "/en-US/docs/Accessibility/ARIA/ARIA_Live_Regions) 属性确保我们的自定义错误信息将呈现给所有人，包括使用屏幕阅读器等辅助技术

##### JavaScript

以下 JavaScript 代码演示如何设置自定义错误校验。

// 有许多方式可以获取 DOM 节点；在此我们获取表单本身和// email 输入框，以及我们将放置错误信息的 span 元素。

var form = document.getElementsByTagName('form')[0];var email = document.getElementById('mail');var error = document.querySelector('.error');

email.addEventListener("input", function (event) {

// 当用户输入信息时，校验 email 字段

if (email.validity.valid) {

// 如果校验通过，清除已显示的错误消息

error.innerHTML = ""; // 重置消息的内容

error.className = "error"; // 重置消息的显示状态

}}, false);

form.addEventListener("submit", function (event) {

// 当用户提交表单时，校验 email 字段

if (!email.validity.valid) {

// 如果校验失败，显示一个自定义错误

error.innerHTML = "I expect an e-mail, darling!";

error.className = "error active";

// 还需要阻止表单提交事件，以取消数据传送

event.preventDefault();

}}, false);

# 有许多库可用于执行表单校验;[Validation](http://bassistance.de/jquery-plugins/jquery-plugin-validation/" \o "http://bassistance.de/jquery-plugins/jquery-plugin-validation/)

你应该毫不犹豫地使用它们。 这里有一些例子：

* 独立的库（原生 Javascript 实现）：
  + [Validate.js](http://rickharrison.github.com/validate.js/" \o "http://rickharrison.github.com/validate.js/)
* jQuery 插件:
  + [Validation](http://bassistance.de/jquery-plugins/jquery-plugin-validation/" \o "http://bassistance.de/jquery-plugins/jquery-plugin-validation/)
  + [Valid8](http://unwrongest.com/projects/valid8/" \o "http://unwrongest.com/projects/valid8/)

#### 远程校验

在某些情况下，执行一些远程校验可能很有用。 当用户输入的数据与存储在应用程序服务器端的附加数据绑定时，这种校验是必要的。 一个应用实例就是注册表单，在这里你需要一个用户名。 为了避免重复，执行一个 AJAX 请求来检查用户名的可用性，要比让先用户发送数据，然后因为表单重复了返回错误信息要好得多。

执行这样的校验需要采取一些预防措施：

* 它要求公开 API 和一些数据；您需要确保它不是敏感数据。
* 网络滞后需要执行异步校验。这需要一些用户界面的工作，以确保如果校验没有适当的执行，用户不会被阻止。