Unlimited WordPress themes, graphics, videos & courses! **From $16.50/m**

tuts+ Subscribe Sign In

Advertisement

Form Input Validation Using Only HTML5 and Regex

by Monty Shokeen 19 Apr 2019 Difficulty: Beginner Length: Medium Languages: English

HTML5 Forms Regular Expressions

Validation of form input is something that should be taken seriously. With luck, nothing worse than garbage data will be submitted to a site which uses data from forms without proper validation. However, there also a chance that hackers will be able to compromise the private data of users who trusted you with their information.

Since validation is so important, it makes sense that there are tools and libraries to validate and sanitize data on both the front-end and the back-end.

In this tutorial, our focus will be on using the built-in features of HTML5 to validate different kinds of input without relying on any external libraries. Obviously, you should not stop at just HTML5-based validation, but this would be a good start to make forms on the website more secure.

The Form Input Element

Whenever you want to get some kind of input from your users, you will most likely use the HTML `input` element. It doesn't matter if you want to know their first name, last name, email address, the city they currently live in, their phone number, or their favorite sports team. The `input` element is a very user-friendly way of getting information from our visitors.

However, some malicious users would like to take advantage of the fact that they can enter almost any kind of string into an input element and submit a

form. Similarly, there might be some users who just didn't know that they are entering the data in the wrong format.

Both these problems can be solved very easily by using some HTML5 attributes with your form elements.

The Type Attribute

The `type` attribute will determine what kind of input is considered valid for a given element. When no value is specified for the `type` attribute, the type is set to `text` by default. This basically means that all kinds of text inputs will be considered valid for that particular element.

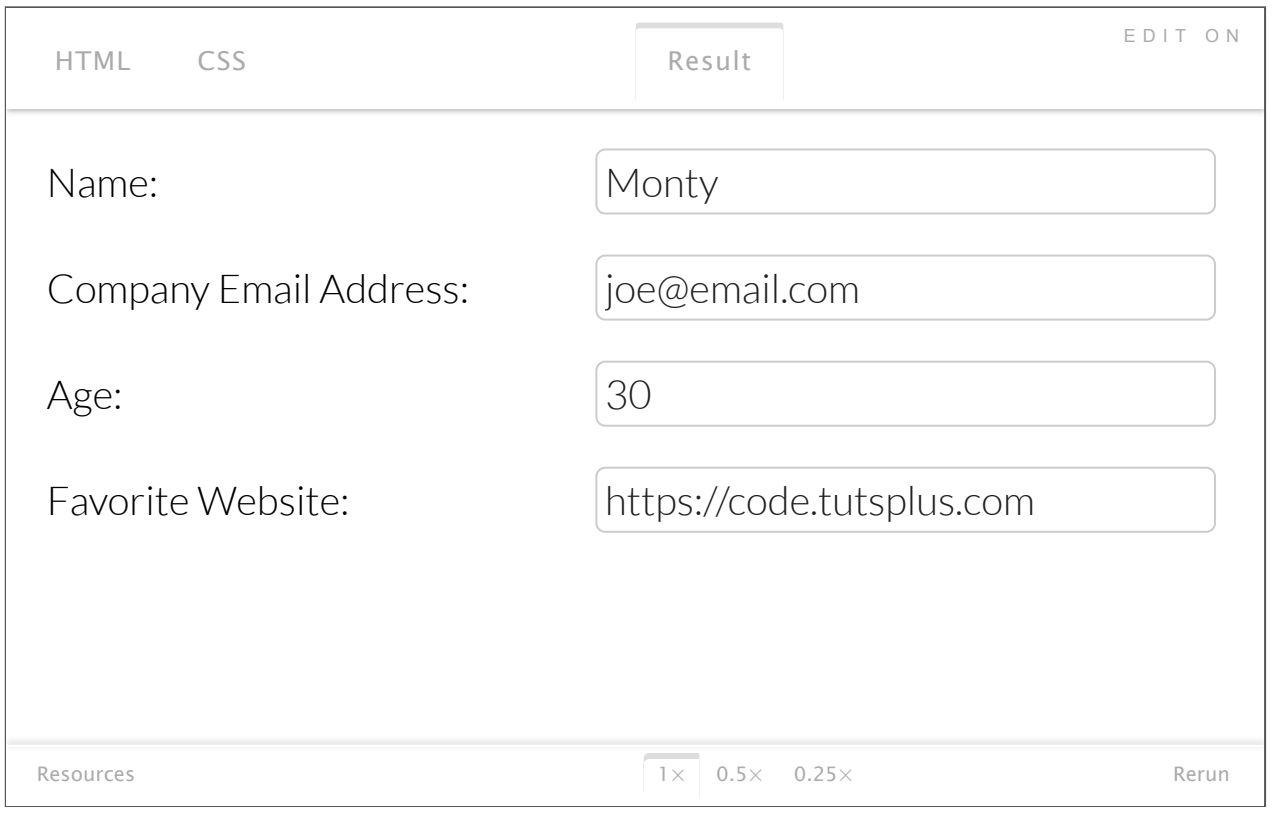
This is fine when you want users to input their names. However, when you want them to enter their email address or numbers like their age and weight, it is much better to set the value of the `type` attribute to something appropriate.

Here are a couple of values that you can choose from:

- `email`: This will ask users to enter their email address in a valid format. For instance, they can't just write **myemail.com** or **something@** or **@something.com**. They will have to enter a value similar to **myemail@domain.tld**. Of course, they can still enter non-existent emails, but that is a different issue!
- `number`: This will make sure that only numbers are considered valid input. For example, when you ask someone their age in a form, they won't be able to submit **potato** or **thirty six** as input. They will have to write an actual number like **36** or **15**.
- `url`: You can set the type attribute to `url` if you want users to enter a valid URL into the input element. This will prevent them from entering something like **tutsplus**. However, **tutsplus.com** will also be considered invalid—users will have to enter a full URL like **https://tutsplus.com**.
- `tel`: Using this value is not as helpful as others because the format for a telephone number varies all over the world. There is just no standard pattern that browsers can match against the input to determine if the number is valid. However, setting the type to `tel` can be helpful at a later stage when you do your own custom validation.

There are many other values for the type attribute which can be used to specify the type of input that is valid for a particular element. You can read about all these values on the Input element reference page on MDN.

The following CodePen demo shows how we can use the type attribute to control what is permitted in different input fields.



Ad

Advertisement

The Minimum and Maximum Length Attributes

One more way to restrict what passes as valid input for an element is to use the `minlength` and `maxlength` attributes. These set the minimum and maximum number of characters that need to be entered in an input element to make it valid.

The right values for both these attributes will vary on a case-by-case basis. For instance, some websites might want a username to be between 4 and 15 characters long, while others might limit the maximum length to 12. Similarly, people in some countries will have unusually short or long names compared to others.

Using Regex for Form Validation

Setting a `type` attribute value certainly helps us in limiting what passes as valid input. However, you can go even further and specify a pattern that a username or email address has to follow in order to be considered valid.

Let's say you want to make sure that usernames are only alphanumeric. This can be done easily with the help of the `pattern` attribute. You just have to set its value to a regex expression which will act as a guideline to determine which input is valid and which is not.

Here are some examples of using regex with the `pattern` attribute.

```
<input type="text" id="uname" name="uname" pattern="[a-zA-Z0-9]+" minlength=
```

The pattern above will keep checking that all the usernames only contain characters from a-z, A-Z, or 0-9. For example, **monty42**, **42monty**, **MON42ty**, and **mon42ty** are all valid usernames in this case, but **monty_42** is invalid.

The `minlength` and `maxlength` attributes will make sure that the username is not too small or too big.

If you want the username to begin with a particular character like an underscore, you can simply add it to the front of the pattern.

```
<input type="text" id="uname" name="uname" pattern="_[a-zA-Z0-9]+" minlength
```

Now, every username which does not begin with _ and contains any characters besides a-z, A-Z, or 0-9 after that will be considered invalid.

I hope this helps in clarifying how we can use the pattern attribute and some regex to limit what is considered valid input even when the `type` attribute is set to `text`.

Advanced Validation With Regex Patterns

You can also use the `pattern` attribute along with other types of input elements like `email` and `url` to restrict what is considered valid. For example, let's say you only want users to enter a URL which is a subdomain of **tutsplus.com**. You can simply set the value of the pattern attribute to `https://.*\.tutsplus.com`. Now, any input like **https://google.com** or **https://envato.com** will be considered invalid. Even if you use a **tutsplus.com** URL that starts with **https://**, that would be invalid because the URL is supposed to start with **https://**.

The same thing can be done with emails. If you want the emails to end with something specific, you can simply use the pattern attribute for that. Here is an example:

```
<input type="email" id="email" pattern=".+@tutsplus\.com|.+@envato\.com">
```

If the above input element is used in a form, users will only be able to enter an email address that ends with **tutsplus.com** or **envato.com**. This means that **hi@gmail.com** or **howdy@something.com** would be invalid.

Check out our JavaScript Regex Cheatsheet for more examples of regular expressions and tips on how to use them.

REGULAR EXPRESSIONS

JavaScript Regex Cheat Sheet

Monty Shokeen

Required Fields and Placeholder Text

While the `required` and `placeholder` attributes are not necessarily related to validation, they can be used to improve the user experience when someone is filling out a form.

Not everyone is willing to share their information with a website. If a form contains ten different input elements, but only five or six of them are required for what you are want to do and the rest are to get extra information, it's a good idea to let the users know.

You can mark certain input fields as required using the `required` attribute, while leaving the optional fields untouched. This will let users know the absolute minimum information they have to provide when filling out a form. It might also increase the number of people who fill out a form because they will know beforehand that filling out all the fields is not absolutely necessary.

The `placeholder` attribute also goes a long way when it comes to making a form more user friendly. For example, if you don't let users know that they have to enter URLs which begin with **https://** and are subdomains of **tutsplus.com**, they might just give up after unsuccessfully filing up the URL field with **something.com** or **code.tutsplus.com**.

In the following example, we have used the pattern, required and placeholder attributes for more granular control over validation and better user experience.

```
<form>
  <label for="name">Name: *</label>
  <input type="text" id="name" name="name" pattern="[a-zA-Z]+" placeholder=
  <br>
  <label for="name">Company Email Address: *</label>
  <input type="email" id="email" name="email" placeholder="joe@company.com"
  <br>
  <label for="name">Age: </label>
  <input type="number" id="age" name="age" min="10" max="80" placeholder="3
  <br>
  <label for="name">Favorite Tuts+ Website: *</label>
  <input type="url" id="website" name="website" pattern="https://.*\.tutspl
</form>
```

Final Thoughts

In this tutorial, we learned how to add basic validation to our forms by simply using HTML and regex. Using the right value for the `type` attribute will force users to enter information in an input field in a certain format. Use of regular expressions with the `pattern` attribute can help us keep the valid input more constrained.

Finally, we learned how to use the `placeholder` attribute to make sure that the forms we create are user friendly and people filling out information don't get frustrated because they don't know the input format we consider valid.

Form validation certainly does not end here. In my next tutorial, you will learn about different form events in jQuery and how to validate forms using a jQuery plugin.

JAVASCRIPT

Best JavaScript Forms of 2019

Monty Shokeen

Advertisement

Monty Shokeen

I am a full-stack developer who also loves to write tutorials in his free time. Other than that, I love learning about new and interesting JavaScript libraries.

FEED LIKE FOLLOW

Weekly email summary

Subscribe below and we'll send you a weekly email summary of all new Code tutorials. Never miss out on learning about the next big thing.

Translations

Envato Tuts+ tutorials are translated into other languages by our community members—you can be involved too!

Translate this post

Powered by