How to validate Email addresses in a Mautic form using JavaScript

Validation is a method to constantly check, verify, authenticate and prove the accuracy of something. *Email Validation* is a method to check, authenticate and prove the accuracy and validity of emails. Emails is one of the most convenient and popular means of communication used by a lot of people - the need to validate emails is very important as it is:

1. Improve the quality of data: By validating emails, you can ascertain the accuracy of data as only accurate and valid emails are collected, a large amount of invalid email addresses means there is a lot to improve.
2. Improves email marketing: Validation of emails improves email marketing as only deliverable emails are being targeted when sending emails out.
3. Leads to increased deliverability: When emails are validated you don't get to deal with issues that arise with sending out incorrect or invalid emails such as high spam complaint rate.
4. Prevents email hard bounce: Email bounces occur when the recipient email address is not correct or the email does not exist. Email validation cleans up the list of emails and removes invalid or incorrect emails.
5. Improves sender's reputation: Validating emails eliminate roadblocks such as incorrect or mistyped emails that may hinder great sender reputations.
6. Saves money: Sending emails out cost money, email validation eliminates unwanted emails leaving only relevant emails and thereby reducing expenses.

Emails can be validated using different methods such as sending emails using java, using Mautic Campaign to validate email addresses etc. For this article, we would look into how we can set up javascript to validate email addresses

on the client-side, it is very fast and a preferred choice by most people, we would see how this can be done in the following sections.

What are the guidelines that should be observed by email validation?

When performing email validation it is very important you take into consideration the email structure. This is an important rule that should be observed. Emails have two parts that are separated by this _@_ symbol. The two parts are called local and domain parts respectively. The first part which is the local part usually consists of:

- Capital letters (A-Z) and small (a-z) letters
- Numerals (0-9)
- Some special characters which are:! # $ % & ‘ * + – / = ? ^ _ ` { | ~
- . (period, dot or full stop) which cannot be the first or last character and cannot be repeated

The second part which is the domain part consists of:

- Capital letters (A-Z) and small (a-z) letters
- Numerals (0-9)
- “.” and “-” dot and hyphens

How to use Regex

When validating emails on the client-side with javascript the most preferred means is to use Regular Expression also known as Regex.

Note: There is no universal or general approved regex expression that is used for email validation. Most people use different regex expressions that satisfy different use cases when validating emails. Before we jump right into using any regex, we would have to test it with some email address (both correct and

incorrect) to be sure it throws an error when a fake or an incorrect address is used. The regex example below has been tested to validate up to 99% email addresses.

```
/^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))@((([[0-9]{1,3}.[
0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}])|(([a-zA-Z-0-9]+.)+[a-zA-Z]{2,}))$/
```

To quickly test our regex expression with some email address we would use this function

```
function testRegex(email) {return /^(([^<>()[]\.,;:s@"]+(.[^<>()[]\.,
;:s@"]+)*)|(".+"))@((([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}])|(
([a-zA-Z-0-9]+.)+[a-zA-Z]{2,}))$/.test(email)}
```

Here is the output when testing with these two email addresses.

- testRegex(jane@.example.com) //true
- testRegex(jane..smith@example.com) //false

Using Javascript with Regex

The code example below shows regex implementation in javascript, the code is not hard to understand, from the code if the email value matches the regex then it returns (This is a valid email address) pop up alert if it does not match, it returns(This is not a valid email address) pop up alert.

index.js

```
function ValidateEmail(inputText){var mailformat = /^(([^<>()[]\.,;:s
@"]+(.[^<>()[]\.,;:s@"]+)*)|(".+"))@((([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,
3}.[0-9]{1,3}])|(([a-zA-Z-0-9]+.)+[a-zA-Z]{2,}))$/;if(inputText.value
.match(mailformat)) {alert("Valid email address!");document.form1.tex
t1.focus();return true;}else {alert("You have entered an invalid emai
```

```
l address!");document.form1.text1.focus();return false;}}
```

Now let's apply this javascript code to a form.

Validating emails in a form using javascript

We are going to create a form which has the email input and use javascript to validate emails inputted in the form when submitted. Let's see how this works.

First, we will create a file and call it index.html and place the code below in it. The index.html will use the javascript code created earlier to validate the emails on the client-side.

index.html

```
<!DOCTYPE html><html lang="en"><head><meta charset="utf-8"><title>Jav
aScript form validation with regex</title><link rel='stylesheet' href
='index.css' type='text/css' />      </head><body onload='document.fo
rm1.text1.focus()'><div class="mail"><h2>Input an email and Submit to
 validate</h2><form name="form1" action="#"> <ul><li><input type='tex
t' name='text1'/></li><li> </li><li class="submit"><input type="submi
t" name="submit" value="Submit" onclick="ValidateEmail(document.form1
.text1)"/></li><li> </li></ul></form></div><script src="index.js"></s
cript></body></html>
```

Next, we create a file called index.css and place the code below in it which is the stylesheet to beautify the form.

index.css

```

```

```

```

```
   li {        list-style-type: none;        font-size: 16pt;    }
 .mail {        margin: auto;        padding-top: 10px;        paddin
g-bottom: 10px;        width: 400px;        border: 1px soild  silver
;    }    .mail h2 {        margin-left: 38px;    }    input {
 font-size: 20pt;    }    input:focus, textarea:focus{        backgro
und-color: #2ef8cf;    }    input submit {        font-size: 12pt;
 }    .rq {        color:  #FF0000;        font-size: 10pt;        }
```

Run the code and view the output on your browser Here is the expected output of the code.

Now to test our code with some emails - testing with this email jane.doe@gmail.com will work and give a pop up of *"Valid email address!"* as seen below.

Inputting this email jane..smith@example.com won't work and would give a pop up of *"You have entered an invalid email address!"*. Here is the output below.

You can try testing with more emails, to see which is valid or invalid

- jane@.example.com
- Jane.example.com
- john12@gmail.com
- smith+consult@hotmail.com
- peter@example.com
- jake@example.co.in
- peter.wash@example.com

- john_smith@example.com
- jane@example.company.in
- .jake@example.com
- peter@example.com.
- jane@example.c
- smith@example.company

Other Considerations

Validating email addresses on the client-side with JavaScript using regex is fun to set up and try out as illustrated with the examples above although not everyone can grab the concept of regex as it can be a bit difficult to understand sometimes and there is no general or universal regex used for validating emails, people tend to write the regex according to their needs and some regex expressions may get rid of some valid email addresses.

To avoid the use of some difficult regex expressions when validating emails and also test and debug your emails to fix issues your emails may encounter before you send them out, you can use Mailtrap.

With Mailtrap, you can check if your emails are properly formatted and are not sent to the spam folder. Also, you won't get rid of valid email addresses which may be eliminated when using regex.

Stay happy sending emails!

SyntaxHighlighter.config.stripBrs=true; SyntaxHighlighter.all();