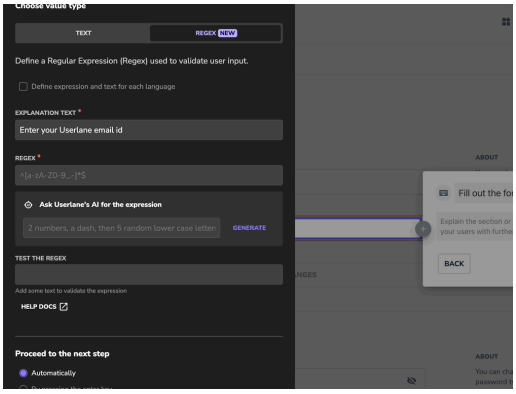
How to use Regular Expressions for Data Validation

Last Modified on 30.06.2023

About Regular Expression

A regular expression (regex) is a sequence of characters that helps you find specific patterns in text. It's commonly used to validate data and ensure its accuracy.



Why use it

Regular expressions can be useful in various scenarios. Here are a few examples:

1. Checking if a text contains your organization's domain.
2. Preventing users from submitting empty form fields.
3. Enforcing a specific format for number fields.
4. Limiting the number of words in a text box (e.g., not more than 10 words).
5. Validating age ranges (e.g., 18-65).

Ultimately, the goal is to collect correct and useful information for your processes.

Example

You want to ensure your colleagues always use the 24-hour format when submitting the expenses of a past business event.

The following regex achieves that:

```
^([01][0-9]|2[0-3]):[0-5][0-9]$
```

In this scenario, the user must use the HH:MM time format in the input field to proceed.

ℹ Important

First, try using the Generate functionality as explained in the help article. Then, use the examples below to improve or adjust the automatically generated Regular Expression to suit your needs.

How to use it

To better understand how basic regular expressions work, let's look at some simple examples:

RegEx	Result	Explanation
abc	abc is allowed	only abc is accepted
Munich|Berlin|Hamburg	The user needs to type in Munich or Berlin, or Hamburg to get the result validated.	"|" is used as "or"
[a-z]	A23BCDeF	At least one small letter from a to z needs to be detected in the string
[0-9]	1user1	The user needs to type in a string that contains at least one number
[0-9]$	31231031231...	It accepts only strings
[0-9] \$$	25$	\ is used to escape a special character. In our case, the $ sign. The second $ shows where the string ends
^[A-Z]	Userlane or USERlane are allowed. uSerlane or userLANE are not allowed	You need to start the string with a Capital letter. It can contain more capital letters.
.	!Userlane#:; is valid	"." is a wild card. If used, any character is accepted in the string.
.{3,}	At least three characters need to be typed in	{3,} means that min 3 characters need to be typed in.
^.{12,20}$	Any string between 12 and 20 characters	{12,20} means the string must contain between 12 and 20 characters. ^ is used to say where the counting starts, while $ shows when the counting ends.
^[1-9]{1}$\|^10$	Validate a number from 1 to 10	^[1-9]{1}$ tells us the first number is 1 to 9. | means or ^10$ tells us that also 10 is accepted
^[1-9]?[0-9]{1}$\|^100$	Validate a number from 1 to 100	The "?" tells us that the preceding character is optional. If it wouldn't be there, then the user would be allowed to type in numbers only from 10 to 100

Now, let's take another look at the example from above:

```
^([01][0-9]|2[0-3]):[0-5][0-9]$
```

The first part of the regular expression *^(([01][0-9]|2[0-3])* tells the user that a value between 00 and 19 is needed or between 20 and 23. In the second part of the regular expression, *[0-5][0-9]$*, a value between 00 and 59 is expected. In that way, only entries in the format of HH:MM are accepted.

Advanced examples

RegEx	Explanation
^[-\w]+(?:\W+[-\w]+){1,4}\W*$	To proceed, type in 1 to 5 words.
^.+?@.+?\..+$	It can be used for email validation. The pattern checks if "@" and "." exist.
((1[0-2]|0?[1-9]):([0-5][0-9]) ?([AaPp][Mm]))$	It validates if the following time format HH:MM 12-hour AM/PM is used.
/^[0-9]{5}([- /]?[0-9]{4})?$/	Checks if the US Postal code was entered correctly.
^(?!01000|99999)(0[1-9]\d{3}|[1-9]\d{4})$	It checks if the entered German Postal Code exists.
^DE[0-9]{20}$	It checks if the German IBAN was correctly entered.