

Regex Text manipulations (RTM) for Invent

There are 4 parts to an RTM:

- 1. Name: This is simply to identify it (the name must be unique).
- 2. Condition Regular Expression: This is a rule that says only apply the replacement if this matchesthe input e.g., you could write a regular expression that matches only number and must be a length of 9 digits, then and only then apply the replacement.
- 3. Replacement Regular Expression: This is where we a capturing the values that we want to replace or add to.
- 4. Replacement String: This is where we specify the replacement that we want to make, replace/remove/add values.

How does it work?

Regex has special characters that allow you to make matches, here are some handy ones:

What is Regular Expression?

Regular Expression (Regex) is a way we can search through text, to do things like validation, get parts of the text, find, and replace and a lot more.

Question mark: ea?: What ever comes before the question mark is optional, in this example this will match e or ea.

The star: ea*: This star characters means its optional like the question mark but also will match as many as possible, in this example it will match e or ea or eaa or eaaa... as may a's as in a row as possible.

The period or full stop:..cat.:: This will matchanything in this example, it will match anything 2 characters before and after cat. If you wanted to match the full stop itself, you can do this: \. (the backslash escapes the special character, just becomes a normal full stop).

The square brackets: [fc]: This will match for any f or c in the input, so if you want to do a match just of letters then you can do [a-zA-Z] or just numbers [0-9].

Parentheses: (t|T): This enables you to make groups, anything in the parentheses will be their own group and will only act a pone themselves. In this example it will search for t or T, if you put some after it will match it e.g. (t|T)he, will match the or The.

Curly brackets: {1,3}: this will match base on length of the characters, in this example its any characters 1 to 3 in a row, if you want just 10 in a row you can use {10}.

Carrot: ^: this will match the beginning of the line, it will always start and only match from the start of your text

Non-Capturing Groups: (?:[0-9]): All this means that the group that we are using in the case numbers 0 to 9, will not be store/captured as we don't need to keep the values. Use case would be in a condition/validation, where we are just checking if some is there.

Digits: \d: This matches any digit character 0-9.

Target your groups: \$1\$2: When you are using the replace you can apply any of your groups by using the dollar and in group number.

Please note that I have just mentioned a handful of special characters, there are a lot more that you can do with Regex.

How can we apply this to an RTM?

Let say that you want to format a phone with the following rules:

- · Apply if we have 10 digests.
- · The format we want is like this: 333.333.3333.

The Condition Regular Expression, this is a check in the example we are checking if its all numbers and its 10 digests long:

```
numbers 0 to 9

Non-

Capturing > ^s*(?:[0-9]s*){10}$ < 10 digits

Groups
```

Then we have the Replacement Regular Expression, where we will capture the groups in this case that we want.

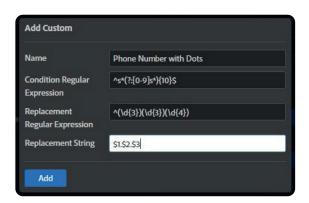
```
Group 2

V
Group 1 > ^(\d{3})(\d{4}) < Group 3
```

Then we have the Replacement Regular Expression, where we will capture the groups in this case that we want.

```
Group 2

V
Group 1 > $1.$2.$3 < Group 3
```



You can test out your regular expression here: https://regexr.com/