Regular Expressions

Bibliography

- Libros y tutoriales:
  - T. Nield, An Introduction to Regular Expressions. O'Reilly Media, Inc. 2021
  - https://docs.oracle.com/javase/tutorial/essential/regex/
  - https://www.tutorialspoint.com/java/java_regular_expressions.htm
- Cheat sheets:
  - https://regexlib.com/CheatSheet.aspx
  - https://cheatography.com/davechild/cheat-sheets/regular-expressions/pdf/

Concept.

- It is a special sequence of characters to match or find other strings or sets of strings, using a specialized syntax held in a pattern. E.g.:
  - Find how many times a string is repeated in a text.
  - Check if a text string has a certain structure.
  - Validate if an email is well written.
  - ...
- Can be used to search, edit, or manipulate text and data.
- Java provides the java.util.regex package for pattern matching with regular expressions. Its classes are Pattern, Matcher and PatternSyntaxException.

Building patterns.

- Metacharacters:

  [] { } () ? * + ^ $. \ = |

- Special characters to build patterns.
- They are non interpreted characters.
- Use:
  - **[] :** set of characters of the expression.
  - **{ } :** lenght of the expresión.
  - **() :** group of a part of a expression.
  - **^:** string starts by.
  - **$:** string end by.
  - \: scape character.

Building patterns. Metacharacters asociated to [ ]

- **? :** 0-1 occurences of the expression.
- *** :** 0-n occurences of the expression.
- **+ :** 1-n occurences of the expression.
- **. :** any character except newline
- **- :** separate values.
- **| :** choose between two values (or).
- **^ :** it does not have

Executing examples

- From this slide, examples of regular expressions can be checked in different ways:
  - Using the java code of **https://docs.oracle.com/javase/tutorial/essential/regex/test_harness.html**
  - Using any online evaluator. E.g.:
    - **http://www.rubular.com/**
    - **https://regex101.com/**
    - **http://www.regexper.com/**

Character Classes

RegEx	Description	Valid	Not Valid
[abc]	a, b, or c (simple class)	a / b /c	g / f / 6
[^abc]	Any character except a, b, or c (negation)	G / h / =	a / b / c
[a-zA-Z]	a through z, or A through Z, inclusive (range)	a / A / H	7 / & / =
[a-zA-Z0-9]	a through z, A through Z, or 0 through 9 inclusive (range)	a / f / H / 6	- / %

Predefined Character Classes

RegEx	Description	Valid	Not Valid
.	Any character	a / : / @	
\d	Digit [0-9]	1 / 2 / 9	a / b / c
\D	Non digit [^0-9]	a / & / =	7 / 4 / 1
\s	whitespace	[\t\n\x0B\f\r]	e / l / q
\S	Non whitespace	d / e / f	[\t\n\x0B\f\r]
\w	Word character [a-zA-Z_0-9]	a / R / 8 / _	& / % / $
\W	Non word character	& / % / $	a / R / 8

Quantifiers

RegEx	Description	Valid	Not Valid
X?	X once	a	ab
X*	X zero or more times	aa	none
X+	X one or more times	aa	" "
X{n}	X, exactly, n times	X{3} aaa	aa / aaaa
X{n, }	X, at least, n times	X{3, } aaaa	a / aa
X{n,m}	X of n to m times	X{3, 5} aaa /aaaa	aa / aaaaaa

Boundary Matchers

RegEx	Description
^	Beginning of a line
\A	Beginning of the input
$	End of a line
\Z	End of the input for the final terminator
\b	Word boundary
\B	Non word boundary

Examples

Description	String	Regular expression
The string matchs exactly with the pattern piat	piat	piat
The string contains piat	Esto es piat de tercer curso	.*piat.*
The string starts by piat	piat es obligatoria	^piat.*
The string ends by piat	Es obligatoria piat	.*piat$
The string starts by piat o Piat	piat	^[pP]iat.*
The string contains only several times p i a t	piatpita	(p|i|a|t)+
La cadena después de una p no va una t	Asigpiat / asigp / Asigptaa	.*p(?!t).*

Examples

Description	String	Regular expression
La cadena no acaba por un digito	Esta cadena tiene 28 digitos	.*[^\d]$
Dirección de correo electrónico	gregorio.rubio@upm.es	^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z09]+)*(\.[A-Za-z]{2,})$
Seleccionar las cadenas de caracteres que están a ambos lados de los caracteres -_.:	piat.sexto_semestre:telematica.ETSIST	split (“[-_.:]”)

Capturing groups

- Parts of the expression can be grouped in parentheses to create capture groups.

Regex: ([a-zA-Z\s]*)(\d+)(.*)

Test line: This order was placed for QT3000! OK?

Groups:

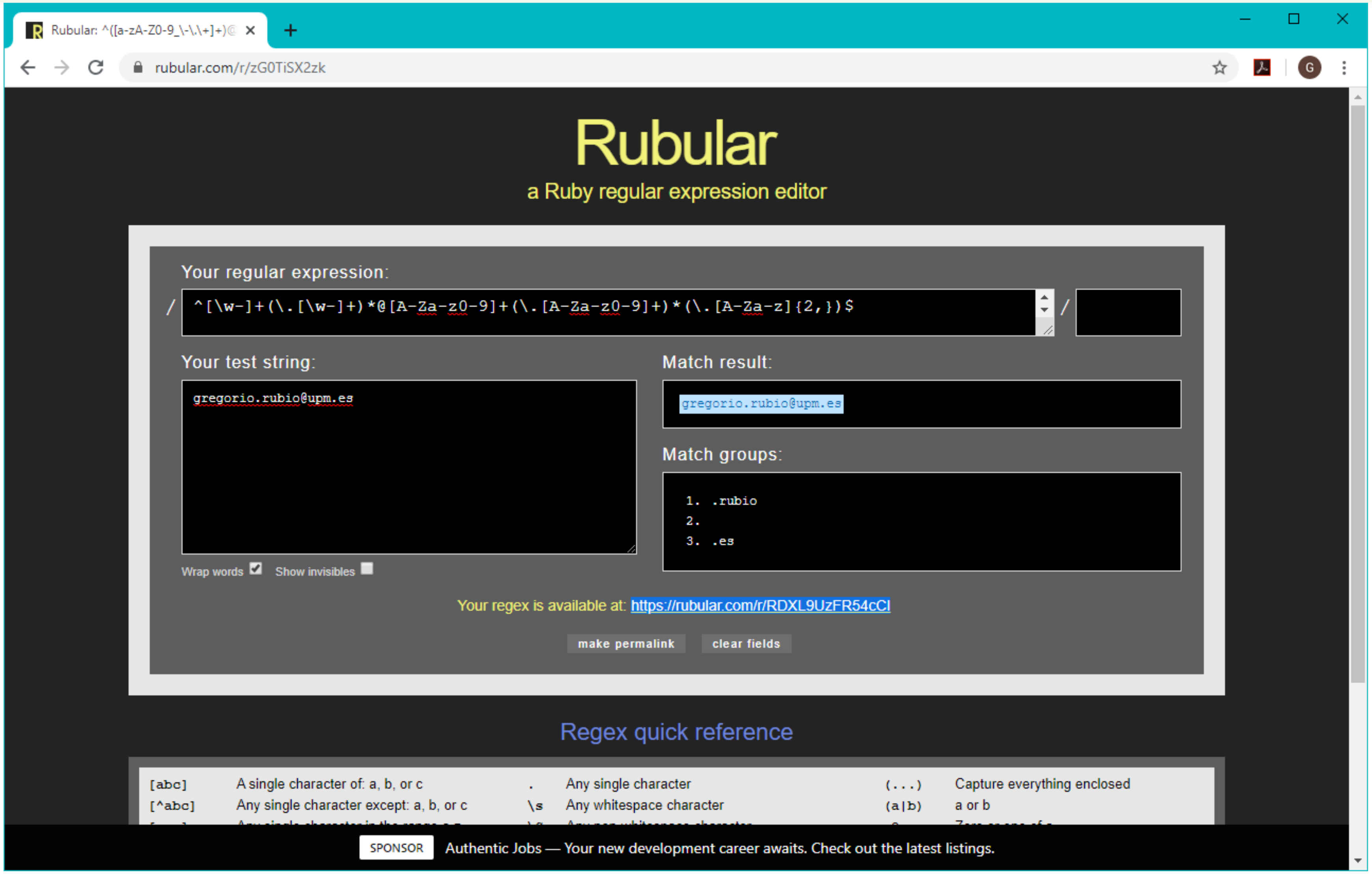
- Group 0: ***This order was placed for QT3000! OK?***
- Group 1: ***This order was placed for QT***
- Group 2: ***3000***
- Group 3: ***! OK?***

Regular Expressions editors and debuggers

- **http://www.rubular.com** : simple evaluator of a regular expression
- **https://regex101.com**: evaluator of a regular expression including a detailed explanation.
- **http://www.regexper.com**: provides a syntax diagram of a regular expression.
- Next three slides provide the result in each editor of the expression that represents the pattern of the e-mail:

^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$

Rubular.com


Rubular: ^([a-zA-Z0-9_\-\.\+]+)@
rubular.com/r/zG0TiSX2zk
Rubular
a Ruby regular expression editor

Your regular expression:

```
/ ^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$ /
```

Your test string:

```
gregorio.rubio@upm.es
```

Match result:

```
gregorio.rubio@upm.es
```

Match groups:

```
1. .rubio
2.
3. .es
```

Wrap words ✓ Show invisibles ☐

Your regex is available at: https://rubular.com/r/RDXL9UzFR54cCI

make permalink
clear fields

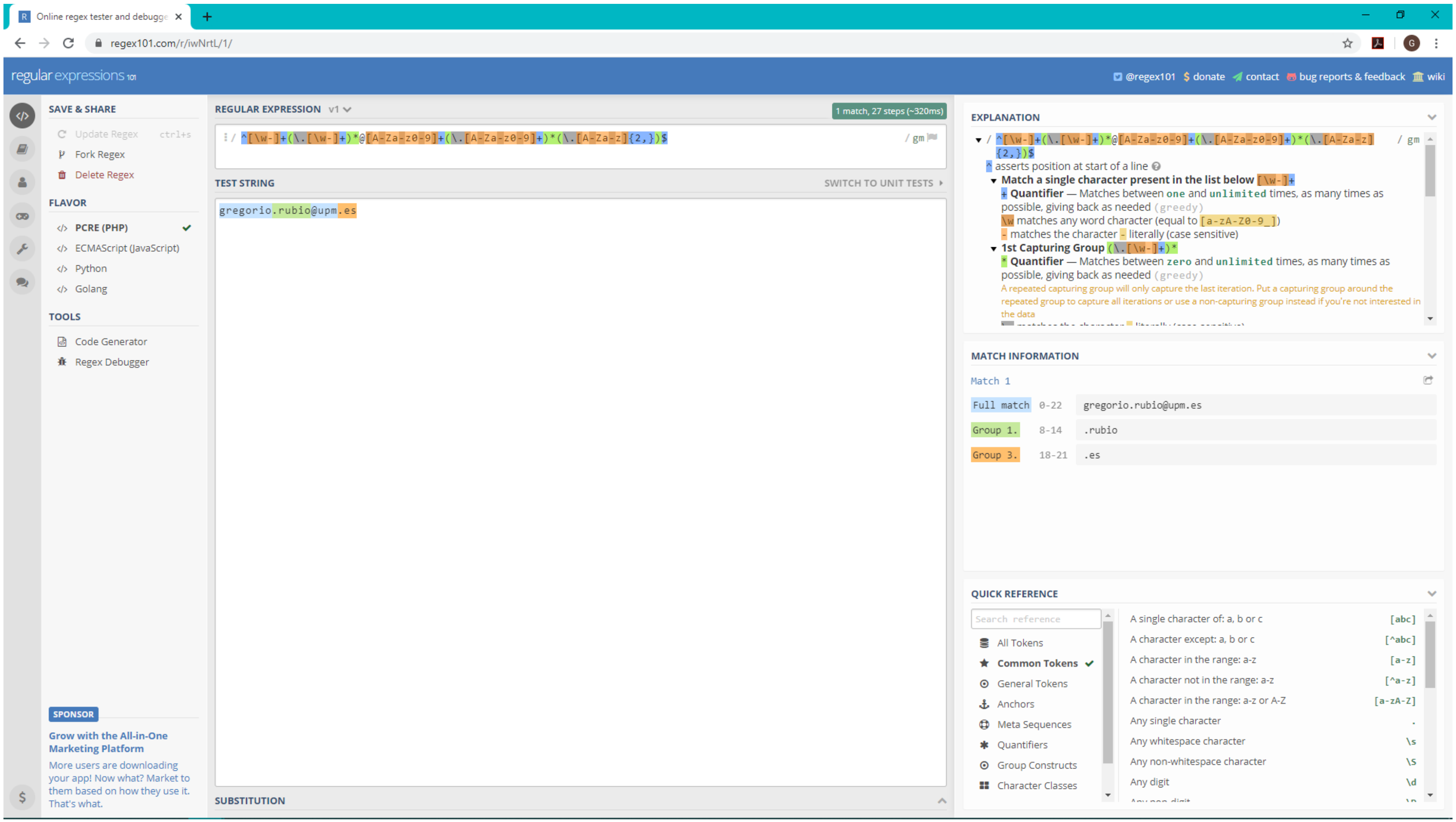
Regex quick reference

[abc]	A single character of: a, b, or c	.	Any single character	(...)	Capture everything enclosed
[^abc]	Any single character except: a, b, or c	\s	Any whitespace character	(a|b)	a or b

SPONSOR Authentic Jobs — Your new development career awaits. Check out the latest listings.

Permalink: https://rubular.com/r/RDXL9UzFR54cCI

Regex101.com



REGULAR EXPRESSION v1 — 1 match, 27 steps (~320ms)

```
/ ^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$ / gm
```

TEST STRING

```
gregorio.rubio@upm.es
```

EXPLANATION

```
/ ^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$ / gm
```

^ asserts position at start of a line

Match a single character present in the list below [\w-]+

+ **Quantifier** — Matches between one and unlimited times, as many times as possible, giving back as needed (greedy)

\w matches any word character (equal to [a-zA-Z0-9_])

- matches the character - literally (case sensitive)

1st Capturing Group (\.[\w-]+)*

* **Quantifier** — Matches between zero and unlimited times, as many times as possible, giving back as needed (greedy)

A repeated capturing group will only capture the last iteration. Put a capturing group around the repeated group to capture all iterations or use a non-capturing group instead if you're not interested in the data

MATCH INFORMATION

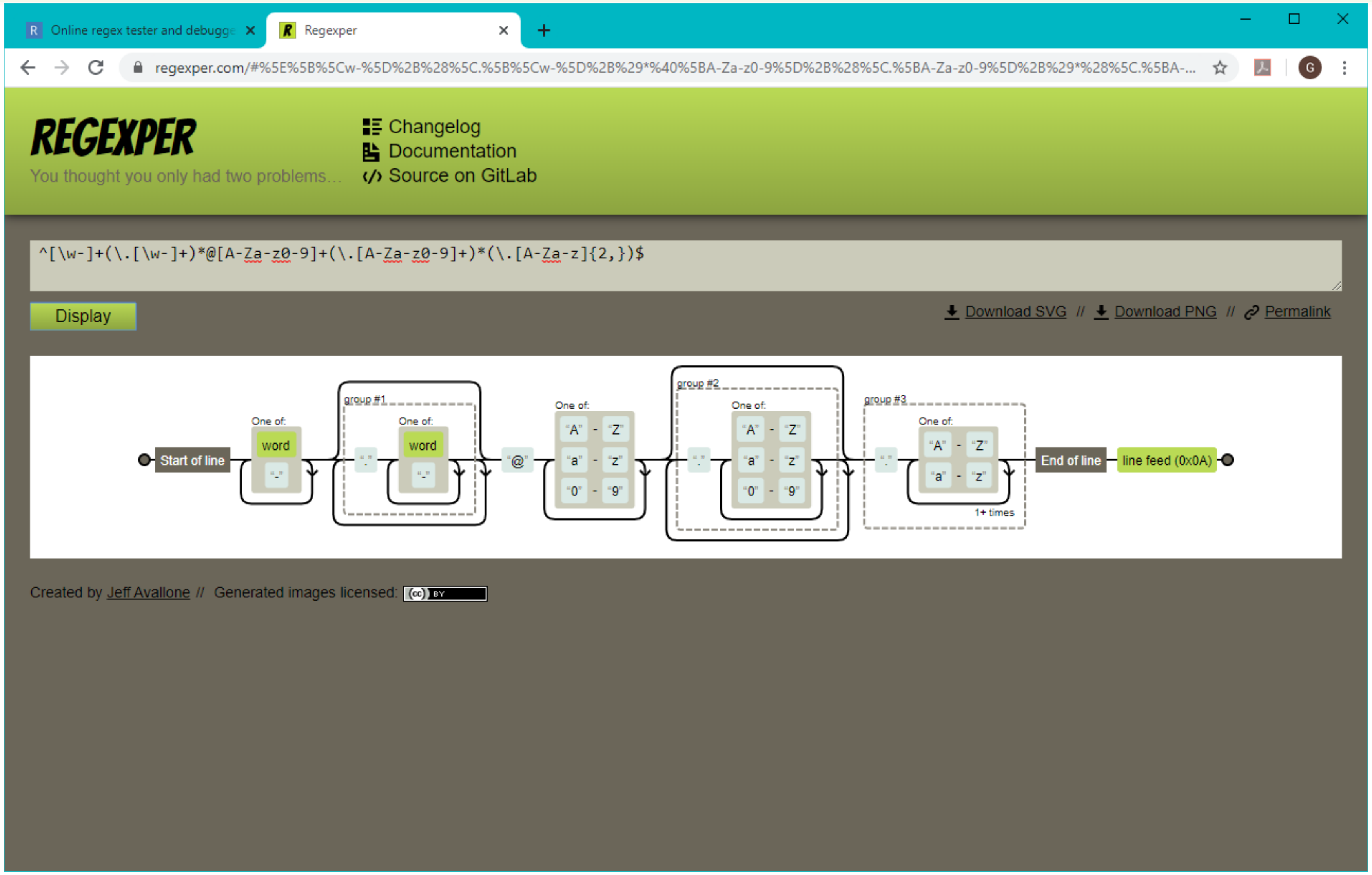
Match 1		
Full match	0-22	gregorio.rubio@upm.es
Group 1.	8-14	.rubio
Group 3.	18-21	.es

QUICK REFERENCE

A single character of: a, b or c	[abc]
A character except: a, b or c	[^abc]
A character in the range: a-z	[a-z]
A character not in the range: a-z	[^a-z]
A character in the range: a-z or A-Z	[a-zA-Z]
Any single character	.
Any whitespace character	\s
Any non-whitespace character	\S
Any digit	\d

Permalink: https://regex101.com/r/iwNrtL/1

Regexper.com


Online regex tester and debugge
Regexper
regexper.com/#%5E%5B%5Cw-%5D%2B%28%5C.%5B%5Cw-%5D%2B%29*%40%5BA-Za-z0-9%5D%2B%28%5C.%5BA-Za-z0-9%5D%2B%29*%28%5C.%5BA-...
REGEXPER
You thought you only had two problems…
Changelog
Documentation
Source on GitLab
Display
Download SVG // Download PNG // Permalink
Start of line
One of:
word
"-"
group #1
"."
"@"
"A" - "Z"
"a" - "z"
"0" - "9"
group #2
group #3
1+ times
End of line
line feed (0x0A)
Created by Jeff Avallone // Generated images licensed:

```
^[\w-]+(\.[\w-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$
```

Permalink: https://regexper.com/#%5E%5B%5Cw-%5D%2B%28%5C.%5B%5Cw-%5D%2B%29*%40%5BA-Za-z0-9%5D%2B%28%5C.%5BA-Za-z0-9%5D%2B%29*%28%5C.%5BA-Za-z%5D%7B2%2C%7D%29%24%0A

Regex and java

- Java provides the `java.util.regex` package for pattern matching with regular expressions.
- The `java.util.regex` package consists of:
  - Classes:
    - `Pattern`
    - `Matcher`
  - Interface:
    - `MatchResult`
  - Exception:
    - `PatternSyntaxException`
- Javadoc:

https://docs.oracle.com/javase/8/docs/api/java/util/regex/package-summary.html

java.util.regex classes

- `Pattern:`
  - It is a compiled representation of a regular expression.
  - The Pattern class provides no public constructors.
  - To create a pattern, you must first invoke one of its public static **`compile()`** methods, which will then return a Pattern object. These methods accept a regular expression as the first argument.
- `Matcher:`
  - It is the engine that interprets the pattern and performs match operations against an input string.
  - The Matcher class, provides no public constructors.
  - You obtain a Matcher object by invoking the **`matcher()`** on a Pattern object.

java.util.regex classes

- `PatternSyntaxException`:
  - A `PatternSyntaxException` object is an unchecked exception that indicates a syntax error in a regular expression pattern.

Pattern class method

- Method: `compile()`. To create a pattern, invoking `public static compile ()` methods. This methods accept a regular expression as the first argument. It will return a `Pattern` object.

  Pattern pattern = Pattern.*compile (regexExp);*

- Compile () method can be parametized.

  Pattern pattern = Pattern.*compile (regexExp,*

  PATTERN.CASE_INSENSITIVE);

- See parameters at

https://docs.oracle.com/javase/tutorial/essential/regex/pattern.html

Pattern class method

- Method: `split()`. To get the text that falls on either side of the regular expression (`splitChars`), in the input sequence (`InputString`).

```
String splitChars = [-_.:];
Pattern pattern = Pattern.compile (splitChars);
String[] items= pattern.split(inputString);
```

- See code example at

https://docs.oracle.com/javase/tutorial/essential/regex/pattern.html

(some) Matcher class methods.

- Index methods:
  - `start()`: return de start index of the match.
  - `end()`: return de offset of the match character matched.
- Study methods:
  - `find()`: find the next subsequence of the input sequence that matches the pattern.
- Replacement methods:
  - `replaceAll()`: replace every subsequence mached in the input sequence with the replacement string given.
- See all methods at https://docs.oracle.com/javase/tutorial/essential/regex/matcher.html

PatternSyntaxException class methods.

- `getDescription()`: return th description of the error.
- `getIndex()`: return the error index.
- `getPattern()`: return the erroneus pattern.
- `getMessage()`: return a multi-line description of the syntax error.

- See all methods at https://docs.oracle.com/javase/tutorial/essential/regex/pse.html

Regular Expressions