

Expresiones regulares - regex

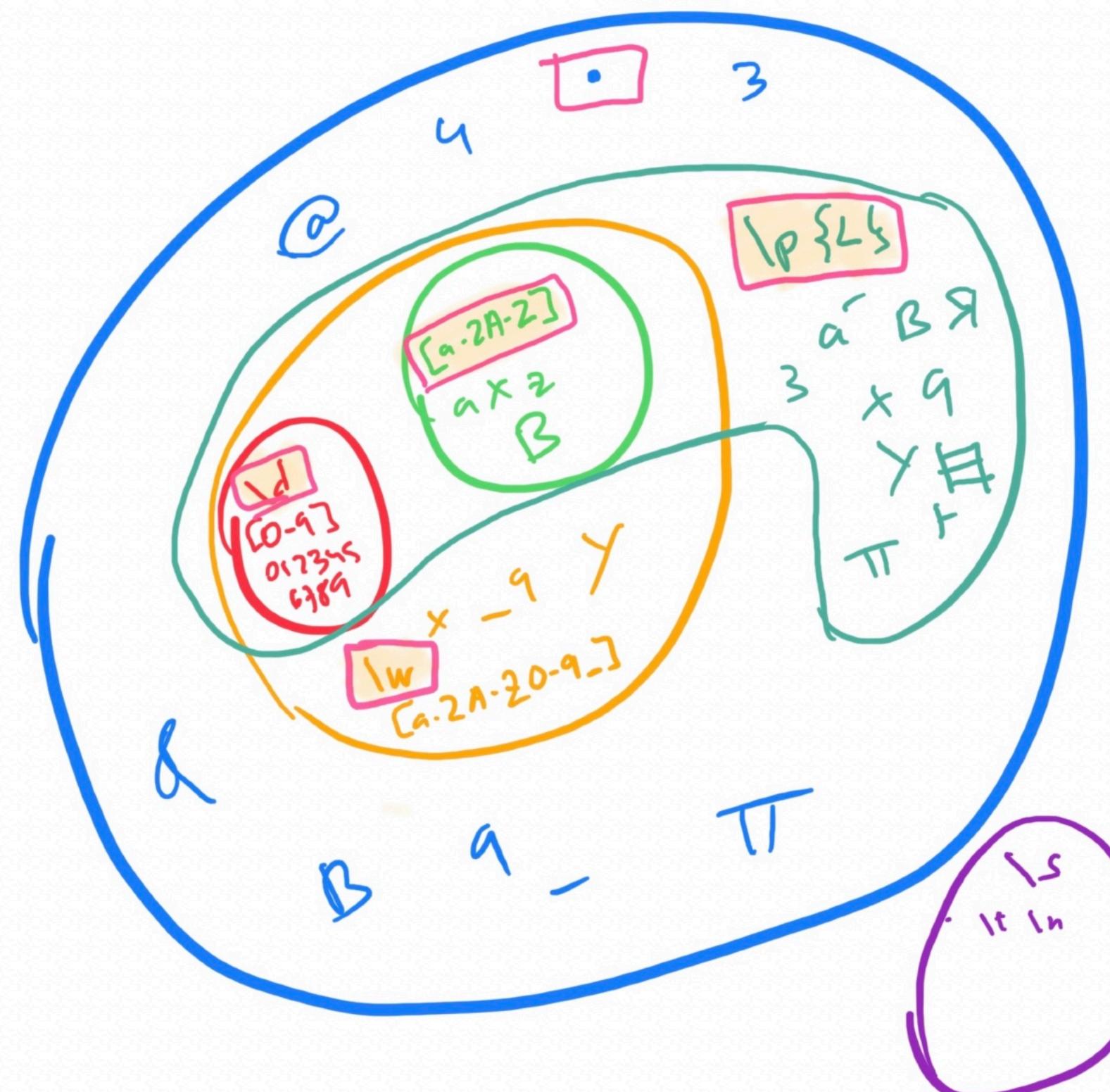
Daniel Illescas - github.com/illescasDaniel

→ Método para encontrar patrones en texto.

- Texto que **empiece** por una palabra: **key .***
- Texto que **termine** en algo: **.*\.\.txt**
- Patrones de caracteres:
 - **Tengo \d+ años**
 - **Me llamo [a-z]+**

Referencia básica

- Letras, números, etc: `abcd 1234`
- Cualquier dígito: `\d`, ej.: `6 , 4` → `[0-9]`. Inverso: `\D`.
- Cualquier letra o número: `\w`, ej.: `a, Z, 4` → `[a-zA-Z0-9_]`. Inv.: `\W`
- Cualquier letra de cualquier idioma: `\p{L}`
- Cualquier carácter (excepto salto línea...): `.`, ej.: `a, 6, &, -`, etc.
- Espacios, tabulación o nueva línea: `\s` → `[\r\n\t\f\v]`. Inv.: `\S`



Referencia básica

- Algún carácter especificado: [] Ej.: [ab5-_%] → 5, b, -, a, _, %
- Un carácter u otro |: a|b → a, b, ab. (red|blue) → red, blue
- Rango caracteres [-]: [4-9], [f-v]

Rápido7, Rapido1, rápid04, rápido0

↓

(R|r)(á|ä) pido [0-9]

• (Rápido | Rapido | rápid0 | rápido)[0-9]

Referencia básica - cuantificadores

- 0 o 1 vez ?: abcd? → abcd, abc - a(100)? → a, a100
- 0 o más *: rápidos* → rápido, rápidos, rápidossss
- 1 o más +: rápidos+ → rápidos, rápidosssssssssssss
- De n a m veces {n, m}: 5{2, 4} → 55, 555, 5555
- Exactamente n veces {3}: 6{3} → 666

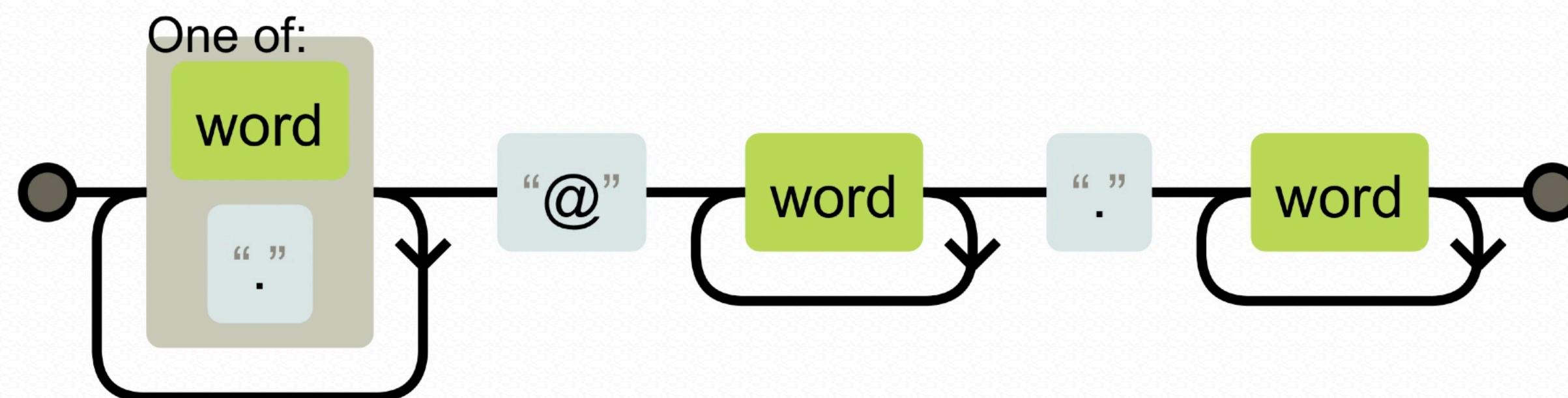
382-555-0112
12-200-0119
4512-201-8472

\d{2-4} - \d{3} - \d{4}
" " - [0-9][0-9][0-9] - "

Validaciones

Tan específicas como quieras

- Validación correo (básica): `[\w\.\.]+@[w+\.]\w+`



<https://regexper.com/>

Website

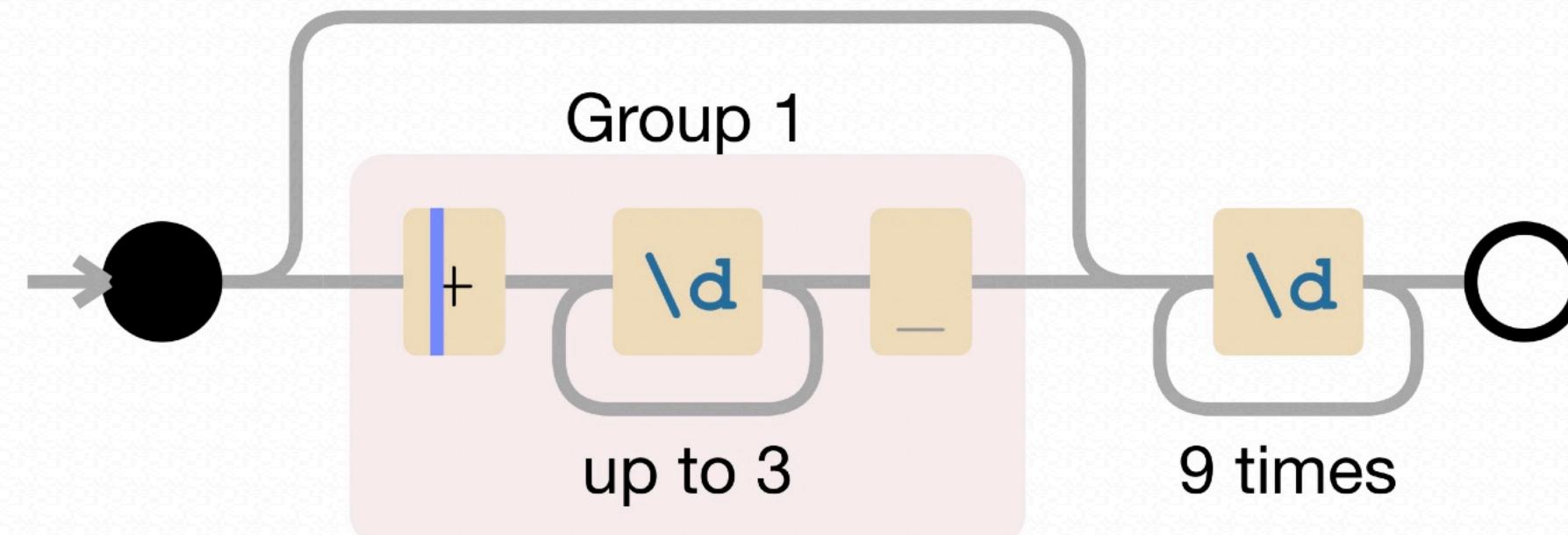
<https://regex101.com/r/VCNpZS/2>

<https://regex101.com/r/VCNpZS/2>

Validaciones

Tan específicas como quieras

- Validación móvil (básica): `(\+\d{1,3})?\d{9}`



<https://www.debuggex.com/>

Website

<https://regex101.com/r/gEm0GK/3>

<https://regex101.com/r/gEm0GK/3>

Validaciones

Tan específicas como quieras

- # ■ Validación correo (avanzada):

```
(?:[a-zA-Z0-9!#$%&'*+/?^_`{|}~-]+(?:\.[a-zA-Z0-9!#$%&'*+/?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-zA-Z0-9](?:[a-zA-Z0-9-]*[a-zA-Z0-9])?\.\.)+[a-zA-Z](?:[a-zA-Z0-9-]*[a-zA-Z])?)|\[(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.\.){3}(?:25[0-5]|2[0-4][0-9]| [0-9][0-9]?)|[a-zA-Z0-9-]*[a-zA-Z](?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\]\)
```

- Validación teléfono (avanzada): \+\\(9[976]\\d|8[987530]\\d|6[987]\\d|5[90]\\d|42\\d|3[875]\\d|2[98654321]\\d|9[8543210]|8[6421]|6[6543210]|5[87654321]|4[987654310]|3[9643210]|2[70]|7|1)\\d{1,14}\\$

<https://emailregex.com/>

Avanzado

Validaciones numéricas

Website

<https://3widgets.com/>

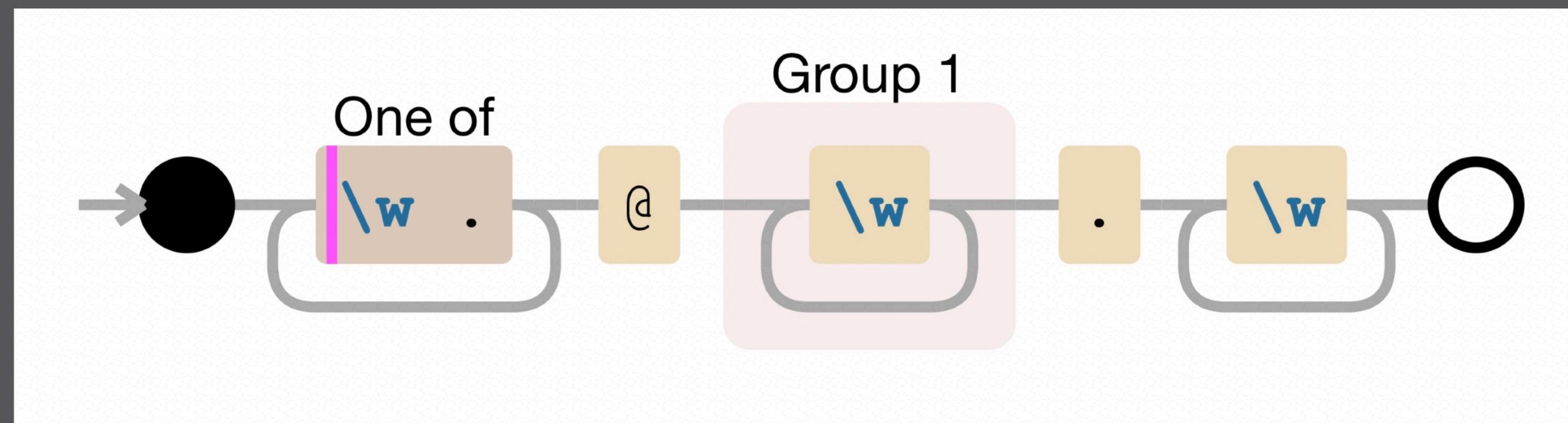
<https://3widgets.com/>

Avanzado

Capturas - *capture groups*

→ Utilizamos paréntesis () para capturar grupos. Sino, (?:).

- Correo capturando el dominio: [\w\.\.]+@(\w+)\.\.\w+



Website

<https://regex101.com/r/lfaD5N/1>

Avanzado

Ejemplos de Sustitución

Website

<https://regex101.com/r/MDNCbF/1>

<https://regex101.com/r/MDNCbF/1>

Avanzado

Ejemplos de Sustitución

Website

<https://regex101.com/r/StMIK7/1>

<https://regex101.com/r/StMIK7/1>

Avanzado

Ejemplos de Sustitución

Website

<https://regex101.com/r/vxFdWI/1>

<https://regex101.com/r/vxFdWI/1>

Avanzado

Ejemplos de Sustitución

Website

<https://regex101.com/r/AGHu9X/1>

<https://regex101.com/r/AGHu9X/1>

Avanzado

Ejemplos en código

```
val basicEmailRegex = Regex("""[\w\.\.]+@\w+\.\w+""")
```

```
val textWithEmail = "send email at: daniel@mail.com"  
val email = "daniel@mail.com"
```

```
basicEmailRegex.containsMatchIn(textWithEmail)  
basicEmailRegex.containsMatchIn(email)
```

```
basicEmailRegex.matches(textWithEmail)  
basicEmailRegex.matches(email)
```

Website

https://pl.kotl.in/t2rJNLt_L

Avanzado

Extraer coincidencias en código

```
val textWithEmail = "send email at: daniel@mail.com or dan@pm.me"  
val textWithEmailMatchResult1 = basicEmailRegex.find(textWithEmail)  
textWithEmailMatchResult1?.value  
textWithEmailMatchResult1?.range  
  
val textWithEmailMatches = basicEmailRegex.findAll(textWithEmail)  
for (matchResult in textWithEmailMatches) { /*...*/ }  
  
val emailMatchResult2 = basicEmailRegex.matchEntire(email)
```

Website

<https://pl.kotl.in/6EqW4MQYy>

Avanzado

Capture groups

```
val basicEmailRegex = Regex("""([\w\.\.]+)@(\w+\.\w+)""")  
val textWithEmail = "send email at: daniel@gmail.com or dan@outlook.com"  
val textWithEmailMatches = basicEmailRegex.findAll(textWithEmail)  
for (match in textWithEmailMatches) {  
    match.groupValues // list  
    for (matchGroup in match.groups) {  
        matchGroup?.value  
        matchGroup?.range  
    }  
}
```

Website

<https://pl.kotl.in/XPaHpSvD8>

Avanzado

Reemplazar coincidencias

```
val telephoneRegex = Regex("""(?:(\+|\d{1,3}) )?(\d{9})""")  
val phonesText = """  
555666777  
+34 888999555  
abcde  
+1 003335553  
666666555  
"""  
  
val output: String = phonesText.replace(telephoneRegex, """phone: ($1) $2""")
```

Website

<https://pl.kotl.in/q8klfgpx0>

Avanzado

Reemplazar coincidencias

```
val output: String = phonesText.replace(telephoneRegex) { match: MatchResult =>
    val groupValues = match.groupValues
    val prefix = groupValues[1]
    val phone = groupValues[2]
    return@replace if (prefix.isEmpty()) {
        "phone: ${phone}"
    } else {
        "phone: (${prefix}) ${phone}"
    }
}
```

Website

https://pl.kotl.in/Vhq_LOkJy

Referencia oficial

Website

<https://www.regular-expressions.info/reference.html>

<https://www.regular-expressions.info/reference.html>

FIN

Website

<https://github.com/illescasDaniel>

