Regular Expressions (part 1)

Stat 133 with Gaston Sanchez

Introduction to Regex

Regex is not ...

a programming language

a markup syntax

a unix utility

Regex is:
a text string that defines
a certain amount of text

Regex is:
a text string that defines
a $\underbrace{\text{certain amount of text}}_{\textbf{pattern}}$

Regex, at its core, has to do with matching patterns of text

Basics of regex

2 main types of characters

Literal Characters

&

Metacharacters

Literal Characters

A literal character is a character that matches itself.

- **Letters** (lower and upper case): a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
- **Numbers**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- **Some symbols:** # ! , ; : % & / = < > @

Metacharacters

A metacharacter is a character that does NOT match itself.

```
.   -   \   +   *   ?   $   ^
    (   )   [   ]   {   }
```

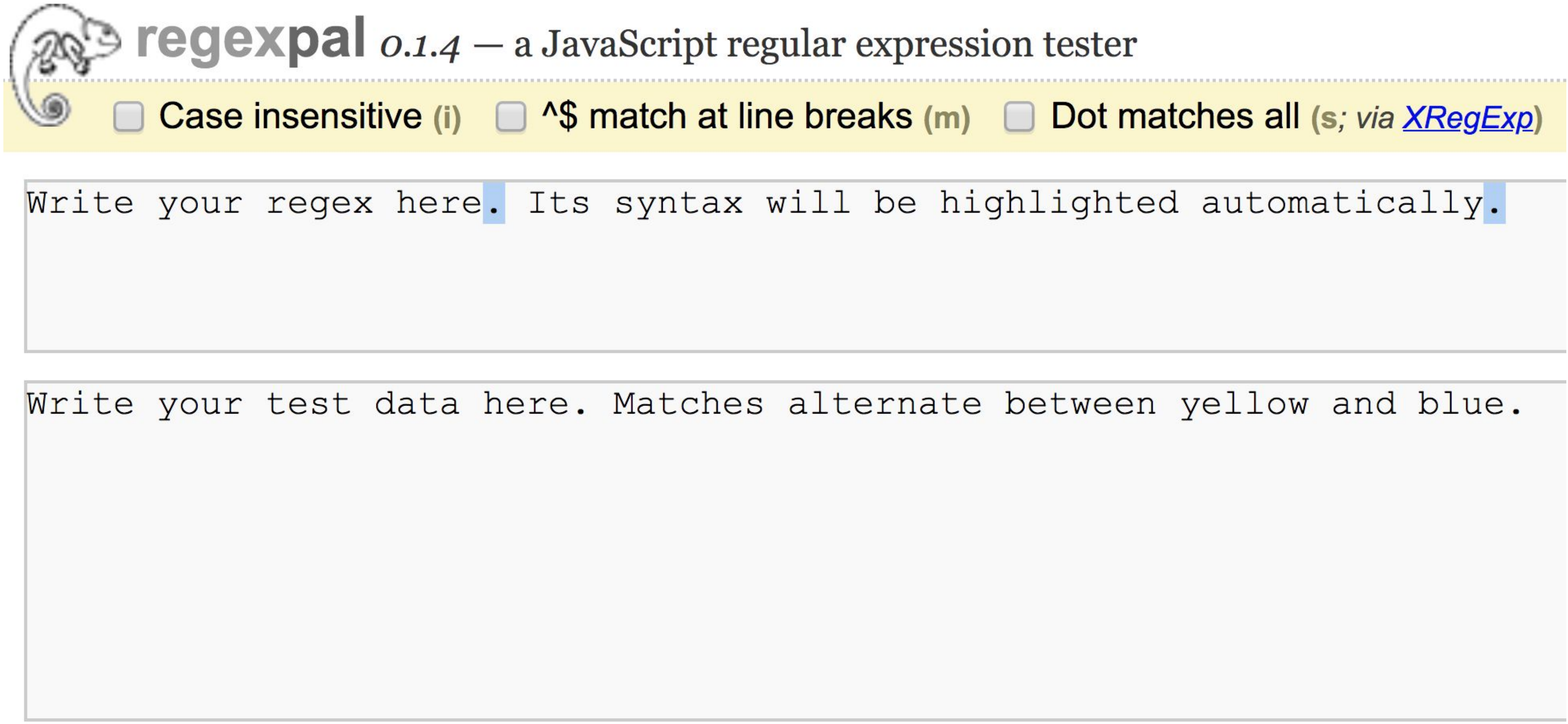
Demo

Let’s use an online regex tester: **regexpal**

We’ll use an old version 0.1.4, good for illustration purposes.

Right now we are covering regular expressions in general. Later, we’ll see what R provides in terms of regex capabilities.

http://regexpal.com.s3-website-us-east-1.amazonaws.com/



regexpal *0.1.4* — a JavaScript regular expression tester

□ Case insensitive (i) □ ^$ match at line breaks (m) □ Dot matches all (s; *via XRegExp*)

```
Write your regex here. Its syntax will be highlighted automatically.
```

```
Write your test data here. Matches alternate between yellow and blue.
```

Examples: literal chars.

car

bike

bus

airplane

train

boat

```
car
bike
bus
airplane
train
boat
```

Regex pattern:

`a`

```
car
bike
bus
airplane
train
boat
```

Regex pattern:

```
i
```

```
car
bike
bus
airplane
train
boat
```

Regex pattern:

`ai`

```
car
bike
bus
airplane
train
boat
```

Regex pattern:

`air`

“Wildcard” (dot) Metachar: .

```
5.00
5100
5 00
5-00
```

```
5.00
5100
5 00
5-00
```

Regex pattern:

.

↑

wildcard metacharacter
“dot”

matches ANY character

```
5.00
5100
5 00
5-00
```

Regex pattern:

```
\.
```

escape metacharacter
“backslash”

converts a metachar into
a literal character

Regex pattern:

```
5.
```

wildcard metacharacter
“dot”

```
5.00
5100
5 00
5-00
```

Regex pattern:

```
5.00
5100
5 00
5-00
```

`.0`

wildcard metacharacter
“dot”

Character Sets

Character Set

A character set is a set of characters defined (grouped) within brackets **[ ]**

```
car
bike
bus
airplane
train
boat
```

```
car
bike
bus
airplane
train
boat
```

Regex pattern:

```
[aeiou]
```

car

bike

bus

airplane

train

boat

Regex pattern:

`[kt]`

```
car
bike
bus
airplane
train
boat
```

Regex pattern:

```
[car]
```

car

bike

bus

airplane

train

boat

Regex pattern:

`car`

This is NOT a character set

car
bike
bus
airplane
train
boat

Regex pattern:

`[^aeiou]`

Negation metachar
“caret”

negative set

Gaston Sanchez

```
car
bike
b.s
.irplane
train
boat
```

Regex pattern:

`[.aeiou]`

The "dot" inside brackets is NOT a metacharacter

Character Ranges

Important Character Sets

A character set is a set of characters defined (grouped) within brackets **[ ]**

[0123456789]

[abcdefghijklmnopqrstuvwxyz]

[ABCDEFGHIJKLMNOPQRSTUVWXYZ]

Character Ranges

Character ranges let you define abbreviations for common character classes

[0123456789] → **[0-9]**

[abcdefghijklmnopqrstuvwxyz] → **[a-z]**

[ABCDE···VWXYZ] → **[A-Z]**

Character Ranges

[012⋯789abc⋯xyz] → **[0-9a-z]**

[012⋯789ABC⋯XYZ] → **[0-9A-Z]**

[abc⋯xyzABC⋯XYZ] → **[a-zA-Z]**

[abcdefg34567UVWXYZ] → **[a-g3-7U-Z]**