



RUMAD

Rutgers Mobile App Development

Bonus Workshop 1: Regex, Overview

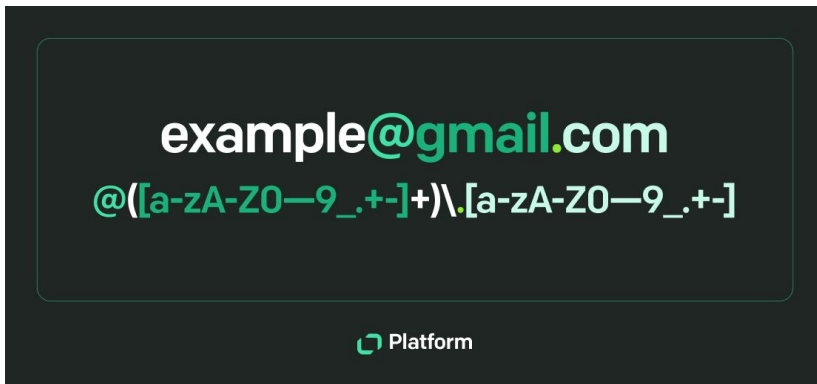
For this bonus workshop, we'll be covering:

- Fundamentals of Regular Expressions
- Its applications
- Implementations in JavaScript

What are Regular Expressions (Regex)?

Simple definition:

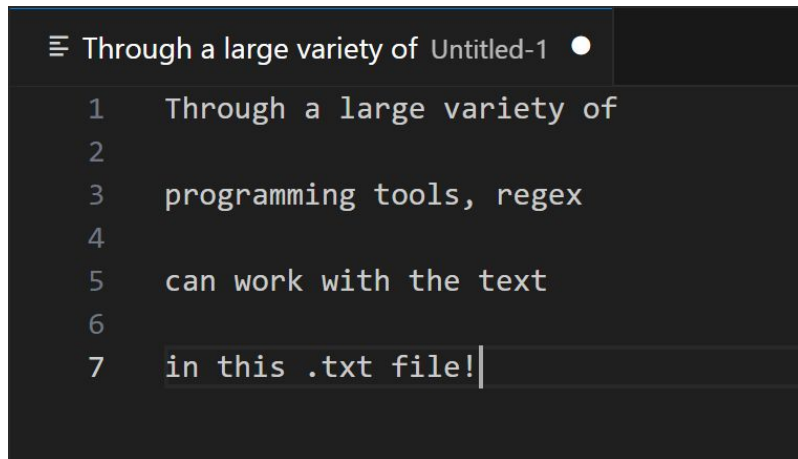
- Sequences of characters used to find and manipulate strings based on patterns



Example of regex, Text Platform

Why do we need RegEx?

- Concise and efficient way to work with text
 - Text processing is extensive in most computational fields
 - Small syntax, large impact
 - Supported by many programming tools

A screenshot of a code editor window titled 'Untitled-1'. The editor has a dark background and shows a text file with the following content:

```
1 Through a large variety of
2
3 programming tools, regex
4
5 can work with the text
6
7 in this .txt file!
```

The text 'in this .txt file!' on line 7 is highlighted with a light blue selection box. The cursor is at the end of the line.

Example .txt file

Applications of RegEx

- Text searching/replacing within files and strings
- Text input validation
 - Ex. emails, passwords, phone numbers
- Data extraction front text
 - Ex. web scraping



Web scraping flow chart, *CrawlNow*

The background is a solid dark red color. Overlaid on this are faint, stylized illustrations of a smartphone. The phone is tilted diagonally. On its screen, there is a satellite dish icon in the upper half and a lightbulb icon in the lower half. The lightbulb has several small dots around it, suggesting it is turned on. The text 'RegEx Symbols' is centered over the phone's screen area.

RegEx Symbols

Symbols

These symbols are paired together to extract/match structured pieces of text.

Symbol	Description	Symbol	Description
^	Start of line +	?	0 or 1 +
\A	Start of string +	{3}	Exactly 3 +
\$	End of line +	{3,}	3 or more +
\Z	End of string +	{3,5}	3, 4 or 5 +
\b	Word boundary +	\	Escape Character +
\B	Not word boundary +	\n	New line +
\<	Start of word	\r	Carriage return +
\>	End of word	\t	Tab +
\s	White space	.	Any character except new line (\n) +
\S	Not white space	(a b)	a or b +
\d	Digit	[abc]	Range (a or b or c) +
\D	Not digit	[^abc]	Not a or b or c +
\w	Word	[0-7]	Digit between 0 and 7 +
\W	Not word	[a-q]	Letter between a and q +
*	0 or more +	[A-Q]	Upper case letter + between A and Q +
+	1 or more +		

Quantity

- $x\{p\}$ matches exactly p repetitions of x
- $x\{p,\}$ matches p or more repetitions of x
- $x\{p,q\}$ matches p to q repetitions of x (Inclusive)
- x^* matches 0 or more repetitions of x
- x^+ matches 1 or more repetitions of x

Quick Examples

$w\{3\}$ matches **www**

$w\{3,4\}$ matches **www**, **wwwwww**

w^* matches "", **w**, **wwwwww**, any number of **w**s

w^+ matches **w**, **wwwwww**, any number of **w** ≥ 1

Grouped Matching

- (cat|dog) matches **cat** or **dog** vs.
[cat|dog] matches 'c', 'a', 't', '|', 'd', 'o', 'g'
- [0-9] matches any digit 0-9 **once**
- [a-zA-Z0-9] matches any alphanumeric
- [^0-9] matches any non-digit character, “^” = **not**

Quick Examples

(cat|dog){2} matches:
catcat, dogdog, catdog, dogcat

[a-zA-Z0-9]* matches:
Cs112 or any word that has
alphanumerics as well as “”

[^a-zA-Z]+ matches:
1 or more combination of digits or
special characters like **1&234\$@**
- Doesn't match ABC123

Shortcuts

- Easy encodings for common matching patterns:
 - alphanumeric:[a-zA-Z0-9]
 - digit [0-9].

For your reference



- **Extra Symbols:**
 - escape character: \
 - . matches any character so .* would match any sequence of characters

Shortcut	Equivalent to
<code>\d</code> <code>\d\d</code> or <code>\d{2}</code>	<code>[0-9]</code> <code>[0-9]{2}</code>
<code>\D</code>	<code>[^0-9]</code> = “NOT” digit
<code>\w</code> <code>\W</code>	<code>[a-zA-Z0-9_]</code> <code>[^a-zA-Z0-9_]</code> = “NOT” word
<code>\s</code> <code>\S</code>	Captures space characters like “ ”, tabs, new-lines, carriage returns, etc. <code>[\t\n\r\f\v]</code> <code>[^ \t\n\r\f\v]</code> = “NOT” space



Examples

Matching Rutgers emails

hpm27@scarletmail.rutgers.edu

What is **constant** in this format?

- The @ symbol
- “.rutgers.edu”

What **varies**?

- The netID, it could be any combination of letters and numbers
- “scarletmail” or could be a different domain such as “rwjms”, “cs”, etc.

Matching Rutgers emails

hpm27@scarletmail.rutgers.edu

So far we have: ____@_____.rutgers.edu

We can match a combination of letters & numbers using `[a-z0-9]+`

- Interpret this as a or b or c... or 0 or 1 or 2...
- The + indicates that we're trying to **match 1 or more**

This gives us: `[a-z0-9]+@[a-z]+\.`rutgers.edu

Gene Matching with Regex

- A genome sequence consists of the letters **A, C, T, G**.
- A potential gene is represented by a string of the form:
 - (prefix) **gene** (postfix)
 - **Prefix** = ATG
 - **Postfix** = TAG/TAA/TGA
 - **Gene** = stuff in between

We want to capture what's in between!

Gene Matching with Regex

So far we have:

ATG____(TAG|TAA|TGA)

What's wrong with this expression?

ATG.*(TAG|TAA|TGA)

- This will match the entire expression including prefix and postfix, what if we want what's in between without extra preprocessing?
- How do we do this? → **Capture Group**

- **Prefix** = ATG
- **Postfix** = TAG/TAA/TGA
- **Gene** = stuff in between

Gene Matching with Regex

- **Prefix** = ATG
- **Postfix** = TAG/TAA/TGA
- **Gene** = stuff in between

To capture what's between the prefix and postfix, we need a **capture group**!

- Surround parts of a pattern string in parentheses to indicate that we want to specifically capture that information

Final Expression:

ATG(.*)(**TAG|TAA|TGA**)

```
const sequence =  
"ATGCATTAG"  
  
const regex =  
/ATG(.*)(TAG|TAA|TGA)/;  
  
const match =  
sequence.match(regex);  
  
//Out:  
// [ATGCATTAG, CAT, TAG]
```




Implementing RegEx in Code

**Code to accomplish examples
from before**

Practice



Break into groups of
3-4 and try the next
few problems!

Question #1

- What does /go+gle/ match?

a) "gogle"

b) "google"

c) "gooogle"

d) All of the above



Question #2

- **Identify all web URL's in the form:**

`https://www.somesite.[com/io/ai]`

-

Question #3

- **Verify all valid dates in the format DD/MM/YYYY.**
Ex: 05/10/25

-

Questions?

Please fill out the feedback form when you have a chance!

Next Bonus Workshop...

Postman and OpenAPI

- Friday 10/24/2025, 3:30 PM–4:30 PM
- Using backend tools, like Postman, to test your APIs
- Getting familiar with OpenAPI standard

Change out QR Code with Link to new
form

Attendance



Please let us know where we can improve the
format of the lessons!