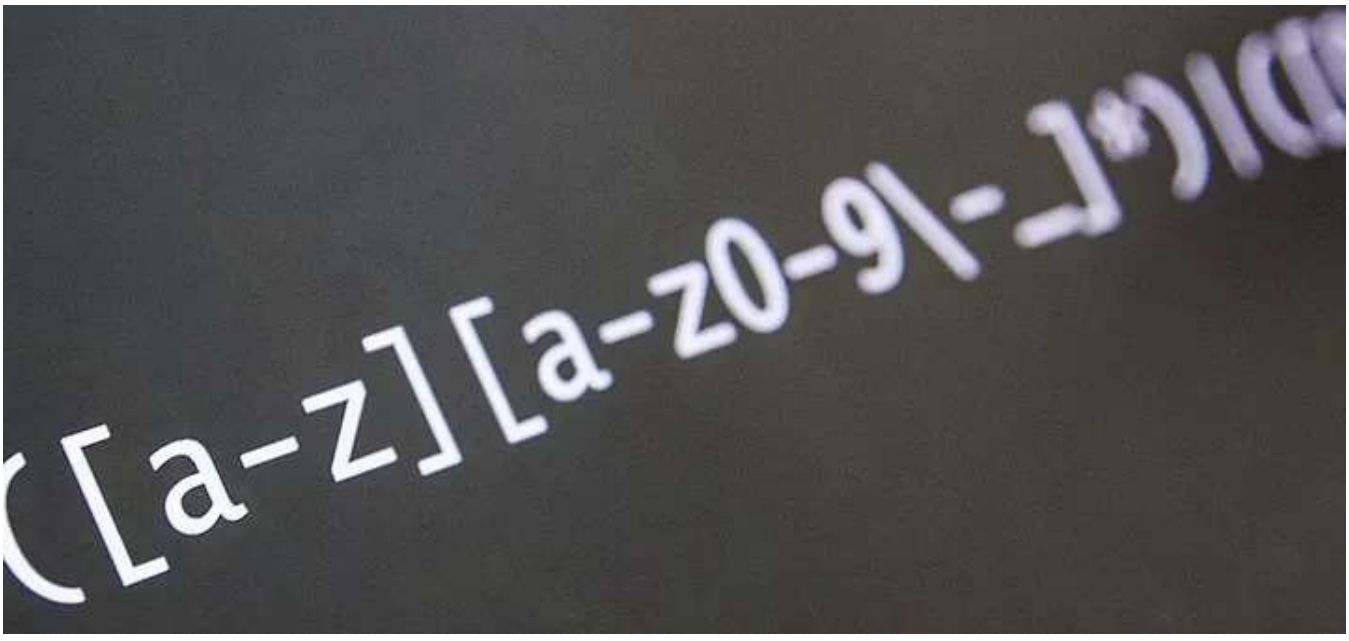
Jonny Fox Follow

Mar 20, 2019 · 4 min read · ✨ · Listen

Save

Regex cookbook — Top 15 Most common regex

The most commonly used (and most wanted) regexes



UPDATE 10/2022: See further explanations/answers in story responses!

After some time I thought about publishing this article (see my previous tutorial) with the regex I used most in the projects on which I was involved.

Write in the comments any regex that you would like to add and I will try (if I can) to implement them 😅

Happy coding!!!

Alpha-numeric, literals, digits, lowercase, uppercase chars only

```
\w                  //alpha-numeric only
[a-zA-Z]            //literals only
\d                  //digits only
[a-z]               //lowercase literal only
[A-Z]               //uppercase literal only
```

Simple numbers — try it!

Matches simple numbers only (no decimal nor fractions)

```
^(\d+)$.            15/12  8.5  12
```

Decimal numbers — try it!

```
^(\d*)[.,](\d+)$    15/12  8.5  12  8,7
```

Fractions — try it!

```
^(\d+)[\/](\d+)$    15/12  8.5  12
```

Alphanumeric without spaces — try it!

```
^(\w*)$             hello123
                    but not this text
```

Alphanumeric with spaces — try it!

Email (simple — see advanced section for more)

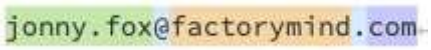
Classic email — try it!

```
^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6})*$
```

```
jonny.fox@factorymind.com
hello@sdasdad.hello
but not this!
```

Email tokens — try it!

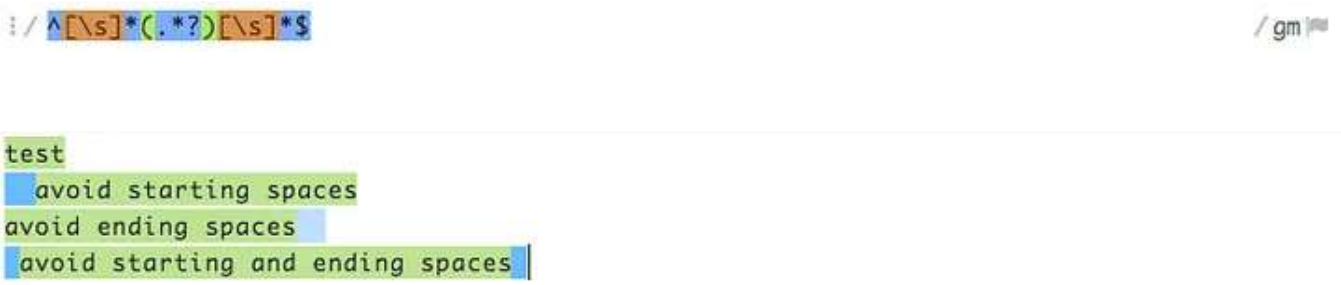
```
^([a-z0-9_\.\+-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$
```



Advanced Regex

Trim spaces — try it!

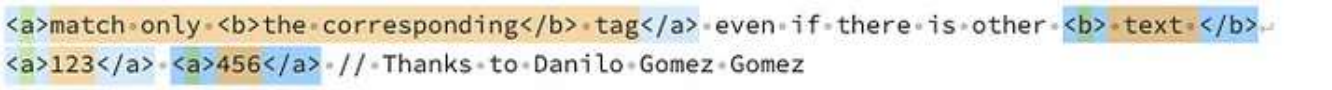
Matches text avoiding additional spaces




```
^[\s]*(.*?)[\s]*$
```

HTML Tag — try it!

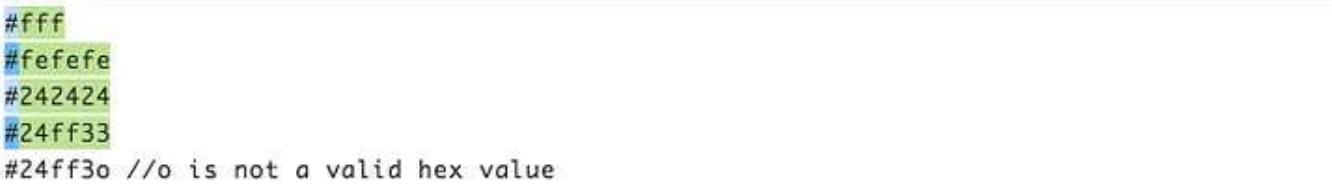
Matches any valid HTML tag and the **corresponding closing tag** (thanks to Danilo Gomez Gomez for the suggestions!)




```
<([a-z]+)[^<]*(?:>(.*?)<\/\1>|\s+\/>)
```

Hexadecimal value — try it!

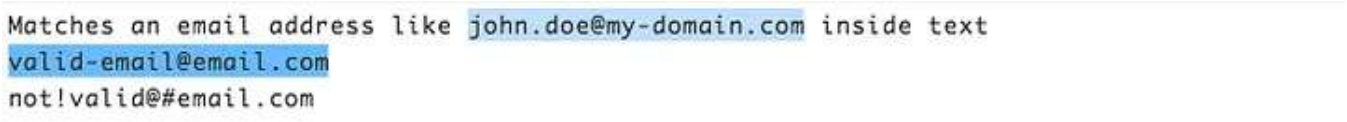
Matches any valid **hex color** inside text




```
\B#(?:[a-fA-F0-9]{6}|[a-fA-F0-9]{3})\b
```

Valid email (RFC5322) — try it!

Matches any valid **email** inside text




```
\b[\w.!#$%&'*+\/=?^`{|}~-]+@[\w-]+(?:\.[\w-]+)*\b
```

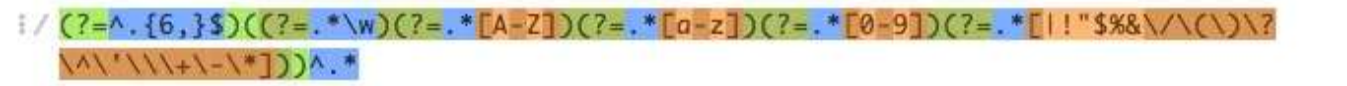
Username (simple) — try it!

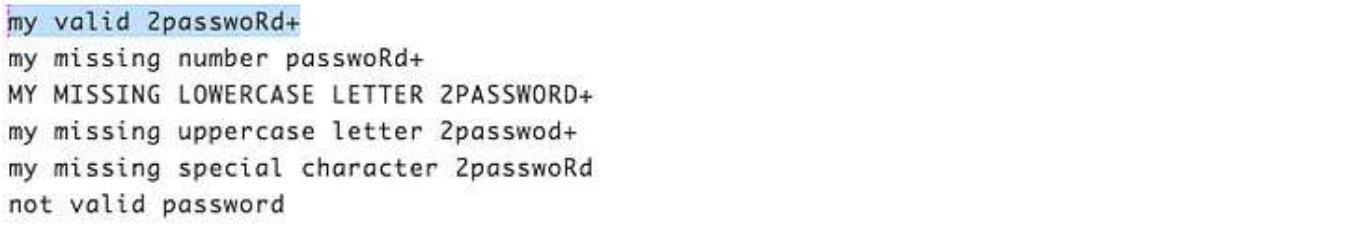
Minimum length of 3, maximum length of 16, composed by letters, numbers or dashes.


```
/^[a-z0-9_-]{3,16}$/
```

Strong password — try it!

Minimum length of 6, at least 1 uppercase letter, at least 1 lowercase letter, at least 1 number, at least 1 special character

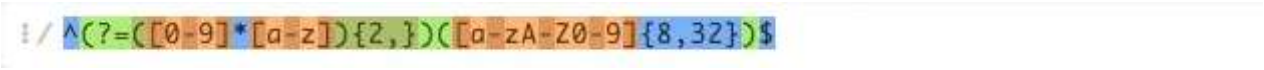




```
(?=^.{6,}$)((?=.*\w)(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])(?=.*[|!"$%&\/\
(\)\?\^\'\\\+\-\*]))^.*
```

2 of a kind — try it!

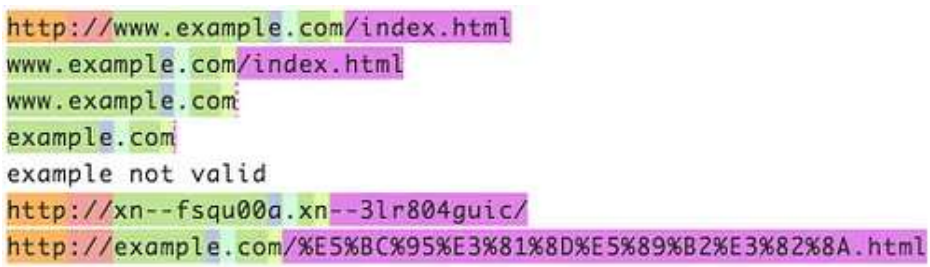
At least 2 letters (uppercase or lowercase) at any index, minimum length of 8, maximum length of 32




```
^(?=([0-9]*[a-z]){2,})([a-zA-Z0-9]{8,32})$
```

URL tokenization— try it!

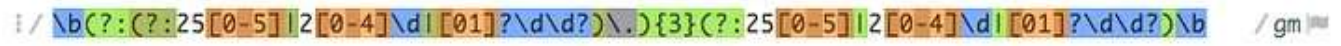
If you want to use capturing groups to get scheme, path, etc. (or add user-info, host, port…) feel free to ask it in comments!

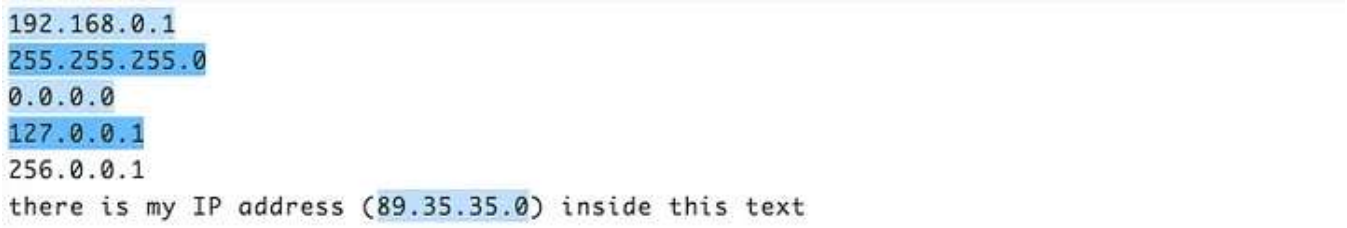


```
^(((https?|ftp):\/\/)?([\w\-\.])+(\.)([\w]){2,4}([\w\/+=%&_\.~?
\-]*))*$
```

IPv4 address — try it!

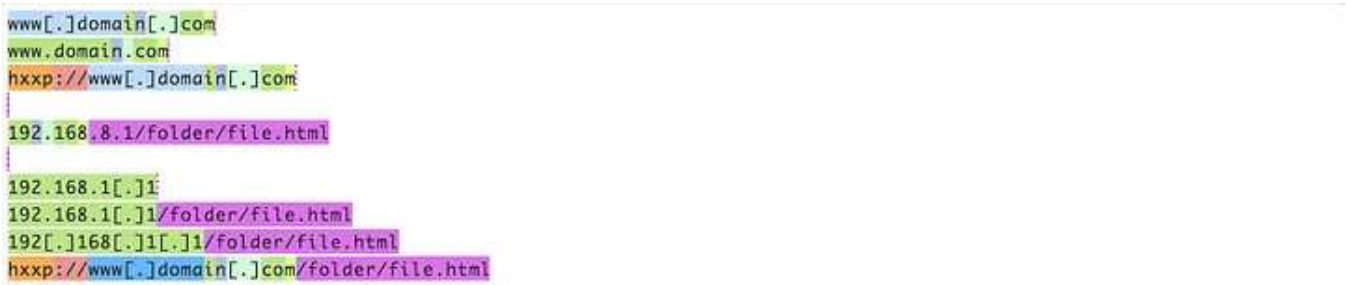
Matches any valid IPv4 address inside text






```
\b(?:(?:25[0-5]|2[0-4]\d|[01]?\d\d?)\.){3}(?:25[0-5]|2[0-4]\d|[01]?
\d\d?)\b
```

URL or IPv4 address — try it!

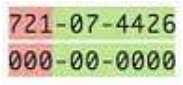



```
^(((h..ps?|f.p):\/\/)?(?:([\w\-\.])+(\[?\.\]?)([\w]){2,4}|(?:
(?:25[0-5]|2[0-4]\d|[01]?\d\d?)\[?\.\]?){3}(?:25[0-5]|2[0-4]\d|[01]?
```

```
\d\d?)))*([\w\/+=%&_\.~?\-]*)$
```

SSN — Social Security Number (simple) — try it!

If you want to check the validity of an SSN feel free to ask in comments!




```
^((?<area>[\d]{3})[-][\d]{2}[-][\d]{4})$
```

Have fun and do not forget to recommend the article if you liked it 💚

Do not forget to check out my previous article about regex tutorial!

Programming Regex Java Python JavaScript