

# Regex

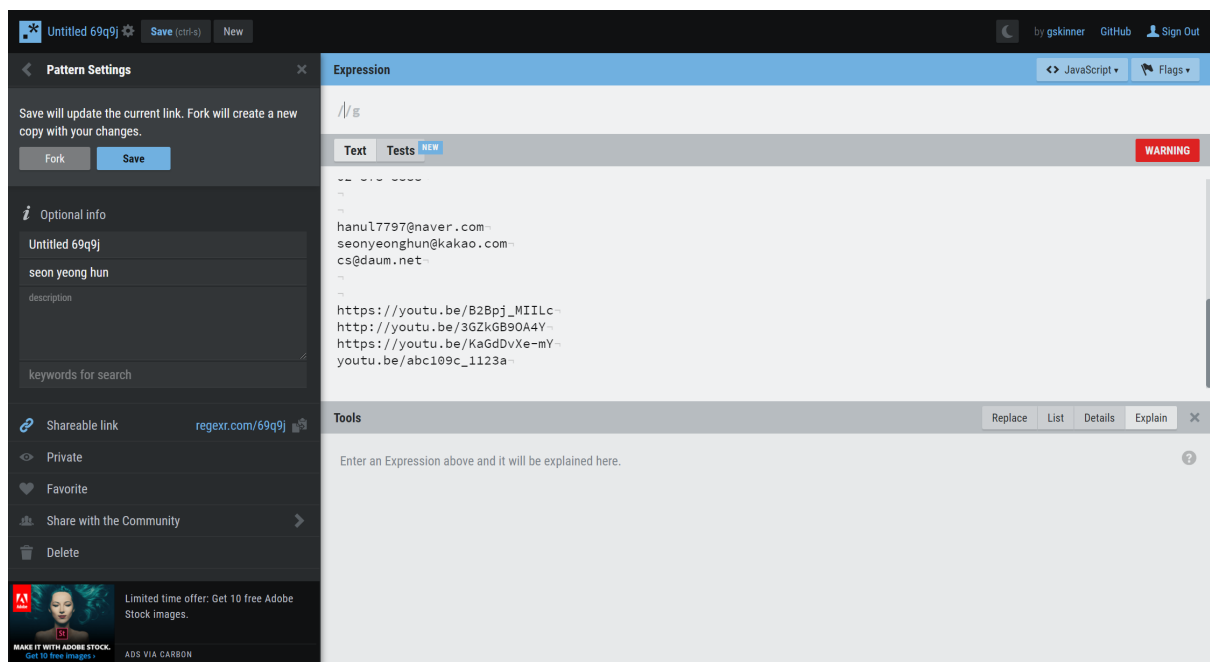
Regular Expression [레직스], 정규 표현식

텍스트에서 우리가 원하는 특정 패턴을 찾을때 요긴하게 사용되는 문법으로, 정규 표현식 또는 정규식이라고 합니다.

특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용하는 형식 언어로, 정규 표현식은 많은 텍스트 편집기와 프로그래밍 언어에서 문자열의 검색과 치환을 위해 지원하고 있습니다.



정규표현식 실습은 <https://regexr.com/69q9j> 에서 하세요!! 연습을 위한 텍스트가 준비되어 있습니다.



## I. 문법정리(pattern)

### 1) Group & Range

문자	뜻
	또는
()	그룹

문자	뜻
[]	문자셋, 괄호안의 어떤 문자든
[^]	부정 문자셋, 괄호안의 어떤 문자가 아닐때
(?:)	찾지만 기억하지는 않음

## 2) Quantifiers

문자	뜻
?	없거나 있거나 (zero or one)
*	없거나 있거나 많거나 (zero or more)
+	하나 또는 많이 (one or more)
{n}	n번 반복
{min,}	최소
{min,max}	최소, 그리고 최대

## 3) Boundary-type

문자	뜻
\b	단어 경계
\B	단어 경계가 아님
^	문장의 시작
\$	문장의 끝

## 4) Character Classes

문자	뜻
\	특수 문자가 아닌 문자
.	어떤 글자 (줄바꿈 문자 제외)
\d	digit 숫자
\D	digit 숫자 아님
\w	word 문자
\W	word 문자 아님
\s	space 공백
\S	space 공백 아님

## II. 연습퀴즈



Quiz1. (예제사이트에서) 전화번호 형식만 선택해보세요

▼ [정답보기]

`\d{2,3}[- .]\d{3}[- .]\d{3,4}`



Quiz2. (예제사이트에서) 이메일 형식만 선택해보세요

▼ [정답보기]

`([a-zA-Z0-9._+-]+)@([a-zA-Z0-9-]+\.)\.[a-zA-Z0-9-]+\.`



Quiz3. (예제사이트에서) 유튜브 아이디 형식만 선택해보세요

▼ [정답보기]

`(?:https?:\/\/)?(?:www\.)?youtu.be\/([a-zA-Z0-9_-]+)`

### III. javascript와 정규식 사용해보기

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>자바스크립트 정규표현식</title>
</head>
<body>
  <h1>정규표현식</h1>
  <h2>자바스크립 with Regex</h2>
  <script>
    var regex = /(?:https?:\/\//)?(?:www\.)?youtu.be\/([a-zA-Z0-9_-]+)/; /* 1. 리터럴 방식 */
    var url = "https://youtu.be/amg3K-mlc";
    <!-- 검사도구, 콘솔 탭을 확인하세요 -->
    console.log(url.match(regex));
    var channel_id = url.match(regex)[1];
    document.write(`당신의 채널 아이디는 ${channel_id} 입니다.`);
    /* var regex = new RegExp(); // 2. 객체생성자 방식 */
  </script>
</body>
</html>
```

# 정규표현식


## 자바스크립트 with Regex

당신의 채널 아이디는 amg3K-mlc 입니다.

### IV. 더 나아가기



<https://regexone.com/> 에서, 정규표현식 퀴즈를 풀어보세요!

 **RegexOne**  
Learn Regular Expressions with simple, interactive exercises.

Interactive Tutorial | References & More

### Lesson 1: An Introduction, and the ABCs

**Regular expressions** are extremely useful in extracting information from text such as code, log files, spreadsheets, or even documents. And while there is a lot of theory behind formal languages, the following lessons and examples will explore the more practical uses of regular expressions so that you can use them as quickly as possible.

The first thing to recognize when using regular expressions is that **everything is essentially a character**, and we are writing patterns to match a specific sequence of characters (also known as a string). Most patterns use normal ASCII, which includes letters, digits, punctuation and other symbols on your keyboard like %#\$@!, but unicode characters can also be used to match any type of international text.

Below are a couple lines of text, notice how the text changes to highlight the matching characters on each line as you type in the input field below. To continue to the next lesson, you will need to use the new syntax and concept introduced in each lesson to write a pattern that matches all the lines provided.

Go ahead and try writing a pattern that matches all three rows, **it may be as simple as the common letters on each line**.

Exercise 1: Matching Characters

Task	Text
Match	abcdefg
Match	abcde
Match	abc

Continue >

*Solve the above task to continue on to the next problem, or read the [Solution](#).*

#### Lesson Notes

- `abc...` Letters
- `123...` Digits
- `\d` Any Digit
- `\D` Any Non-digit character
- `.` Any Character
- `\.` Period
- `[abc]` Only a, b, or c
- `[^abc]` Not a, b, nor c
- `[a-z]` Characters a to z
- `[0-9]` Numbers 0 to 9
- `\w` Any Alphanumeric character
- `\W` Any Non-alphanumeric character
- `{m}` m Repetitions
- `{m,n}` m to n Repetitions
- `*` Zero or more repetitions
- `+` One or more repetitions
- `?` Optional character
- `\s` Any Whitespace
- `\S` Any Non-whitespace character
- `^...$` Starts and ends
- `(...)` Capture Group
- `(a(bc))` Capture Sub-group
- `(*)` Capture all
- `(abc|def)` Matches abc or def