검색만으로는 자바스크립트로 HTML 태그를 지우기 위한 정보를 찾기에 부족해서, 별도의 정규 표현식 공부가 필요했다.
https://blog.naver.com/hyeonjun7/222858261256


박현준 님,
알고리즘 퀴즈칩 베타 테스터로 선정되신 것을 축하드립니다.

엘리스 알고리즘 퀴즈칩 베타 테스터 선정 | [Python] if __na…
엘리스 알고리즘 퀴즈칩 베타 테스터 선정 엘리스라는 코딩 교육 플랫폼이…
blog.naver.com

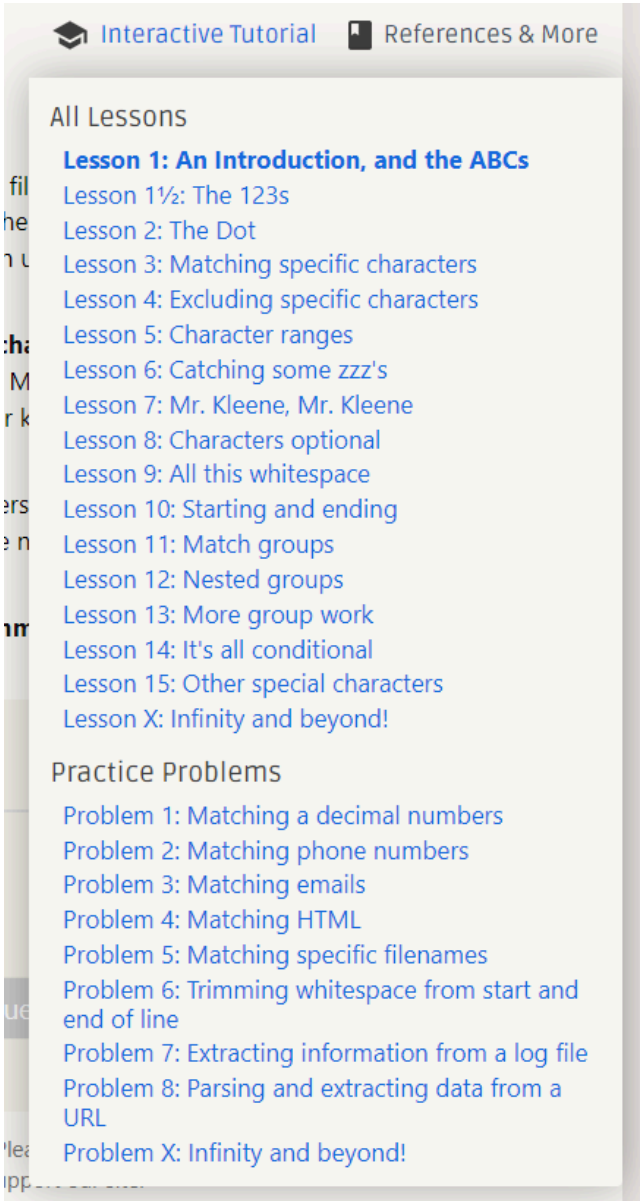
찾아보니 간단한 연습 문제를 풀면서 정규 표현식을 공부할 수 있는 RegexOne(https://regexone.com)이라는 사이트가 있었다.
오늘은 해당 사이트에서 공부한 내용을 정리해보려고 한다.
시작에 앞서, 정규 표현식이란 무엇인가?

## 正規表現式 / regular expression

프로그래밍에서 문자열을 다룰 때, 문자열의 일정한 패턴을 표현하는 일종의 형식 언어를 말한다. 정규식이라고도 부르며,보통 regex 혹은 regexp라 많이 쓴다.

https://namu.wiki/w/%EC%A0%95%EA%B7%9C%20%ED%91%9C%ED%98%84%EC%8B%9D#toc
최근 수정 시각: 2022-05-01 10:17:47

정규 표현식은 문자열 처리가 필요한 상황에서 자주 쓰일 뿐만 아니라 자바스크립트, 파이썬, 루비, 자바, PHP 등 다양한 프로그래밍 언어에 지원되는 형식 언어(formal language)이므로, 공부해두면 유용하게 사용할 수 있다.




RegexOne의 튜토리얼은 기초 수업 15개, 심화 문제 풀이 8개로 구성되어 있다.
RegexOne에서는 정규 표현식의 패턴만 다루기 때문에 자바스크립트에서 사용할 때 알아야 하는 정규 표현식의 형식과 플래그는 따로 찾아 정리하였다.

# [JavaScript] 정규 표현식의 형식과 플래그

자바스크립트에서 정규 표현식의 형식은 다음과 같다.

1. 정규표현식 형식

***/패턴/플래그***

- 슬래시(/) "사이"에는 매칭시킬 "패턴"을 써준다.
- 슬래시(/) "다음"에는 옵션을 설정하는 "플래그"를 써준다.

(플래그는 하나만 찾을지, 모두 다 찾을지 등을 설정하는 옵션이라고 보면 됩니다.)

패턴은 밑에서 자세히 다룰 것이다.
여기서는 플래그의 종류와 용례를 정리해보려고 한다.
용례의 이해가 어렵다면 패턴부터 학습 후 읽어보는 것을 추천한다.

## global search (g)

문자열 맨 앞에서 시작해 가장 먼저 찾은 패턴을 포함해 존재하는 모든 패턴을 찾는다.

```
regex1 = /무야호/
regex2 = /무야호/g
s = '무야호무호야무야호'
res1 = s.replace(regex1, '')
res2 = s.replace(regex2, '')
console.log('without g flag: ' + s + ' → ' + res1)
console.log('with g flag: ' + s + ' → ' + res2)
// console
// without g flag: 무야호무호야무야호 → 무호야무야호
// with g flag: 무야호무호야무야호 → 무호야
```

## case insensitive (i)

패턴의 대소문자를 구분하지 않는다.

```
regex1 = /Dog/
regex2 = /Dog/i
s = 'Surrender is dog'
res1 = s.replace(regex1, 'God')
res2 = s.replace(regex2, 'God')
console.log('without i flag: ' + s + ' → ' + res1)
console.log('with i flag: ' + s + ' → ' + res2)
// console
// without i flag: Surrender is dog → Surrender is dog
// with i flag: Surrender is dog → Surrender is God
```

## multiline (m)

시작 문자(^)와 끝 문자($)가 문자열의 시작과 끝이 아닌 각 줄(개행 문자를 기준으로 구분)의 시작과 끝에서 동작하는 것으로 처리된다.

```
regex1 = /^\d/g
regex2 = /^\d/gm
s = '1초라도 안 보이면\n2렇게 초조한데\n3초는 어떻게 기다려'
res1 = s.replace(regex1, '')
res2 = s.replace(regex2, '')
console.log('without m flag:\n' + s + '\n↓\n' + res1)
console.log('with m flag:\n' + s + '\n↓\n' + res2)
// console
// without m flag:
// 1초라도 안 보이면
// 2렇게 초조한데
// 3초는 어떻게 기다려
// ↓
// 초라도 안 보이면
// 2렇게 초조한데
// 3초는 어떻게 기다려"
// with m flag:
// 1초라도 안 보이면
// 2렇게 초조한데
// 3초는 어떻게 기다려
// ↓
// 초라도 안 보이면
// 렇게 초조한데
// 초는 어떻게 기다려"
```

## unicode (u)

4바이트 문자가 2바이트 문자 두 개가 아닌 4바이트 단일 문자로 처리된다.
유니코드 속성을 사용할 수 있다.
유니코드 속성은 다음과 같은 형태로 사용한다.

```
\p{...}
```

유니코드 속성을 통해 특정 언어, 특수 문자(인용 부호, 통화) 등을 기준으로 단어를 찾을 수 있다.
용례로 쓰인 'Letter'는 모든 언어로 된 문자를 지칭한다. 약자인 'L'로 대체해도 무방하다.

```
regex1 = /\p{Letter}/g
regex2 = /\p{Letter}/gu
str = "ㄱㄴㄷㄹㅁㅃ";
a = str.match(regex1)
b = str.match(regex2)
console.log('without u flag: ' + a)
console.log('with u flag: ' + b)
// console
// without u flag: null
// with u flag: ㄱ,ㄴ,ㄷ,ㄹ,ㅁ,ㅃ
```

Letter 이외에도 많은 유니코드 속성이 존재하는데, 참고할 만한 링크를 아래 몇 개 첨부한다.
https://blog.actorsfit.com/a?ID=01500-aed83fc4-4189-4102-adc4-d756beaa842b
https://ko.javascript.info/regexp-unicode
https://www.regular-expressions.info/unicode.html

## sticky (y)

대소문자를 구분하지 않는다.

## dot all (s)

대소문자를 구분하지 않는다.
만약 패턴에 맞는 문자열인지 확인하고자 하는 목적이라면 아래와 같이 내장함수인 test()를 이용하면 된다.
참고로 아래의 예제는 전화번호 형식의 문자열인
플래그는 여러 개를 사용할 수 있다.
**unicode** (u) Treat the regular expression as a sequence of Unicode code points
**sticky** (y) Matches only from the index indicated by the lastIndex property of this regular expression
**dot all** make . match newline too. Dot metacharacter matches all characters, including newlines

# [JavaScript] 정규 표현식의 패턴

정규 표현식에서 문자들이 어떻게 쓰이는 지 정리한 Lession Notes를 번역하고, 복잡한 식은 시각화한 예제를 덧붙여 개념 정리를 해보려고 한다.
공부를 시작한 목적에 맞게, 자바스크립트를 사용하여 정규 표현식으로 특정 문자열을 선택하고 제거하는 예제를 사용하였다.
만약 패턴에 맞는 문자열인지 확인하고자 하는 목적이라면 아래와 같이 내장함수인 test()를 이용하면 된다.
참고로 아래의 예제는 전화번호 형식의 문자열인지 판별하는 소스 코드이다.
regex는 regular expression의 줄임말.

```
regex = /\d{3}-\d{4}-\d{4}/
t1 = regex.test('010-1111-2222')
t2 = regex.test('01-11-22')
console.log(t1)
console.log(t2)
// console
true
false
```

## abc…

알파벳(a-z)는 해당 알파벳을 나타낸다.

```
s1 = "abcdefg"
s2 = "abcde"
s3 = "abc"
target = /c/
s4 = s1.replace(target, '')
s5 = s2.replace(target, '')
s6 = s3.replace(target, '')
console.log(s1 + ' → ' + s4)
console.log(s2 + ' → ' + s5)
console.log(s3 + ' → ' + s6)
// console
abcdefg → abdefg
abcde → abde
abc → ab
```

123… 숫자(0-9)
₩d 아무 숫자
₩D 숫자가 아닌 모든 문자
. 모든 문자
₩. 마침표(.)
[abc] a 또는 b 또는 c
[^abc] a 또는 b 또는 c가 아닌 모든 문자
[a-z] a부터 z 사이의 문자
[0-9] 0부터 9 사이의 문자
₩w 알파벳과 숫자, 밑줄 문자(_)
₩W 알파벳과 숫자, 밑줄 문자가 아닌 모든 문자
{m} 앞 문자 그룹 m회 반복
{m,n} 앞 문자 그룹 n 이상 m 이하로 반복
* 앞 문자 그룹 0회 이상 반복
+ 앞 문자 그룹 1회 이상 반복
? 앞 문자 그룹은 선택(있어도 되고 없어도 됨)
₩s 모든 공백(탭, 스페이스 등)
₩S 공백이 아닌 모든 문자
^…$ '^' 뒤 문자 그룹으로 시작해서 '$' 앞 문자 그룹으로 끝나는 문자열
(…) Capture Group
(a(bc)) Capture Sub-group
(.*) Capture all
(abc|def) Matches abc or def

(작성중)