# 정규표현식(Regular Expression)

정규 표현식(正規表現式, 영어: Regular Expression, 간단히 RegExp 이라한다.
한마디로 정규식(正規式)은 특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용하는 형식 언어이다. 정규 표현식은 많은 텍스트 편집기와 프로그래밍 언어에서 문자열의 검색과 치환을 위해 지원하고 있다.

문자열에서 문자 조합에 일치 시키기 위하여 사용되는 형식 언어이다. 주로 검색, 치환, 추출을 위한 패턴을 표현하는 데 사용된다.

## 정규표현식

패턴

정규 표현식이라는 문구는 일치하는 텍스트가 준수해야 하는 "패턴"을 표현하기 위해 특정한 표준의 텍스트 신택스를 의미하기 위해 사용된다. 정규 표현식의 각 문자(즉, 패턴을 기술하는 문자열 안의 각 문자)는 메타 문자(특별한 의미로)로 이해되거나 정규 문자('문자 그대로', 즉 '리터럴'의 의미로)로 이해된다.

### 수량의 지정

| | |
|---|---|
| ? | 물음표는 0번 또는 1차례까지의 발생을 의미한다. 이를테면 colou?r는 "color"와 "colour"를 둘 다 일치시킨다. |
| * | 별표는 0번 이상의 발생을 의미한다. 이를테면 ab*c는 "ac", "abc", "abbc", "abbbc" 등을 일치시킨다. |
| + | 덧셈 기호는 1번 이상의 발생을 의미한다. 이를테면 ab+c는 "abc", "abbc", "abbbc" 등을 일치시키지만 "ac"는 일치시키지 않는다. |
| {n} | 정확히 n 번만큼 일치시킨다 |
| {min,} | "min"번 이상만큼 일치시킨다. |
| {min,max} | 적어도 "min"번만큼 일치시키지만 "max"번을 초과하여 일치시키지는 않는다. |

표준 문법

| 메타문자 | 기능 | 설명 |
|---|---|---|
| . | 문자 | 1개의 문자와 일치한다. 단일행 모드에서는 새줄 문자를 제외한다. |
| [ ] | 문자 클래스 | "["과 "]" 사이의 문자 중 하나를 선택한다. "¦"를 여러 개 쓴 것과 같은 의미이다. 예를 들면 [abc]d는 ad, bd, cd를 뜻한다. 또한, "-" 기호와 함께 쓰면 범위를 지정할 수 있다. "[a-z]"는 a부터 z까지 중 하나, "[1-9]"는 1부터 9까지 중의 하나를 의미한다. |
| [^ ] | 부정 | 문자 클래스 안의 문자를 제외한 나머지를 선택한다. 예를 들면 [^abc]d는 ad, bd, cd는 포함하지 않고 ed, fd 등을 포함한다. [^a-z]는 알파벳 소문자로 시작하지 않는 모든 문자를 의미한다. |
| ^ | 처음 | 문자열이나 행의 처음을 의미한다. |
| $ | 끝 | 문자열이나 행의 끝을 의미한다. |
| ( ) | 하위식 | 여러 식을 하나로 묶을 수 있다. "abc¦adc"와 "a(b¦d)c"는 같은 의미를 가진다. |
| ₩n | 일치하는 n번째 패턴 | 일치하는 패턴들 중 n번째를 선택하며, 여기에서 n은 1에서 9 중 하나가 올 수 있다. |
| * | 0회 이상 | 0개 이상의 문자를 포함한다. "a*b"는 "b", "ab", "aab", "aaab"를 포함한다. |
| + | 0 또는 1회 | "a+b"는 "ab", "aab", "aaab"를 포함하지만 "b"는 포함하지 않는다. |
| ? | 0 또는 1회 | "a?b"는 "b", "ab"를 포함한다. |
| ¦ | 선택 | 여러 식 중에서 하나를 선택한다. 예를 들어, "abc¦adc"는 abc와 adc 문자열을 모두 포함한다. |
| {m, n} | m회 이상 n회 이하 | "a{1,3}b"는 "ab", "aab", "aaab"를 포함하지만, "b"나 "aaaab"는 포함하지 않는다. |

| POSIX | ASCII | 설명 |
|---|---|---|
| [:alnum:] | [A-Za-z0-9] | 영숫자 |
| | [A-Za-z0-9_] | 영숫자 + "_" |
| | [^A-Za-z0-9_] | 낱말이 아닌 문자 |
| [:alpha:] | [A-Za-z] | 알파벳 문자 |
| [:blank:] | [ \t] | 공백과 탭 |
| | (?<=\W)(?=\w)\|(?<=\w)(?=\W) | 낱말 경계 |
| [:cntrl:] | [\x00-\x1F\x7F] | 제어 문자 |
| [:digit:] | [0-9] | 숫자 |
| | [^0-9] | 숫자가 아닌 문자 |

| | | |
|---|---|---|
| [:graph:] | [\x21-\x7E] | 보이는 문자 |
| [:lower:] | [a-z] | 소문자 |
| [:print:] | [\x20-\x7E] | 보이는 문자 및 공백 문자 |
| [:punct:] | [][!"#$%&'()*+,./:;<=>?@\^_`{\|}~-] | 구두점 |
| [:space:] | [ \t\r\n\v\f] | 공백 문자 |
| | [^ \t\r\n\v\f] | 공백이 아닌 모든 문자 |
| [:upper:] | [A-Z] | 대문자 |
| [:xdigit:] | [A-Fa-f0-9] | 16진수 |

```
<script>
//정규표현식 메소드

/*
[1].exec();  //필터기능 추출정보를 결과를 배열처리
[2].test();
[3].match();
[4].search();
[5].replace();
[6].split();
*/

var result;
var reg;
var txt;
//1-1. .exec(); 추출 필터 기능
//아래와같이 정규표현식 만들고 ...
// 한글영문대소문자숫자 reg = /([ㄱ-ㅎㅏ-ㅣ가-힣A-Za-z]+)([0-9]+)/g;

// [A-Za-z]괄호 패턴 : 영문 대소문자 A~Z, a~z까지 부분 패턴을 의미한다.
// [a-z]괄호 패턴 : 영문 소문자 a~z까지 부분 패턴을 의미한다.
// ([a-z]+) 괄호패턴 뒤 +(특수문자 Plus기호) 는 반드시 1글자 이상을 연속 사용의 의미한다.
// ([0-9]+)([a-z]+)/g 두개의 (숫자)(문자)패턴 뒤 /g : 전체의 의미( Global의 의미)
// ([0-9]+)([a-z]+)/gi 두개의 (숫자)(문자)패턴 뒤 /gi : 전체의 의미와 대소문자 구분 안함을 의미
                                                                    //i : UpperCase,
LowerCase Insensitive  검색(Search)
//reg = /([A-Za-z]+)([0-9]+)/g;  //패턴순서 문자(영문대소문자), 숫자
reg = /([a-z]+)([0-9]+)/gi;          //UpperCase, LowerCase Insensitive  검색(Search)
```

```
txt ='Age39, Tel025565611, Birth0929';


        ////1회실행 : Age39
        //아래와 같이 .exec() 메소드 구현
        //정규표현식 .exec( txt );
        result = reg.exec( txt );
        //결과 >>
        console.log( result[0] );     //Age39
        console.log( result[1] ); //Age
        console.log( result[2] ); //39



        //2회실행 : Tel025565611
        //아래와 같이 .exec() 메소드 구현
        //정규표현식 .exec( txt );
        result = reg.exec( txt );
        //결과 >>
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );



        //3회실행 : Birth0929
        //아래와 같이 .exec() 메소드 구현
        //정규표현식 .exec( txt );
        result = reg.exec( txt );
        //결과 >>
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );



//1-2. .exec(); 추출 필터 기능
//아래와같이 정규표현식 만들고 ...
reg = /([0-9]+)([A-Za-z]+)/g;      //대소문자 구별없이, 패턴순서 숫자, 문자
reg = /([0-9]+)([a-z]+)/gi;        //대소문자 구별없이, 패턴순서 숫자, 문자
txt ='39Age, 025565611Tel, 0929Birth'; //데이터도 숫자문자형식

        ////1회실행 : Age39
```

```
        //아래와 같이 .exec() 메소드 구현
        //정규표현식 .exec( txt );
        result = reg.exec( txt );
        //결과 >>
        console.log( result[0] );     //Age39
        console.log( result[1] ); //Age
        console.log( result[2] ); //39


        //2회실행 : Tel025565611
        //아래와 같이 .exec() 메소드 구현
        //정규표현식 .exec( txt );
        result = reg.exec( txt );
        //결과 >>
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );


        //3회실행 : Birth0929
        //아래와 같이 .exec() 메소드 구현
        //정규표현식 .exec( txt );
        result = reg.exec( txt );
        //결과 >>
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );


//1-3. .exec(); 추출 필터 기능
var exp = /([A-Za-z]+)([0-9]+)([A-Za-z]+)/g;
var str ='Meber001MoonJong, Meber002JangGo, Meber003SooGa';

        //1회실행
        var answer = exp.exec( str );
                console.log( answer[0] ); //Meber001MoonJong
                console.log( answer[1] ); //Meber
                console.log( answer[2] ); //00
                console.log( answer[3] ); //MoonJong


        //2회실행
        var answer = exp.exec( str );
```

```
        console.log( answer[0] );
        console.log( answer[1] );
        console.log( answer[2] );
        console.log( answer[3] );


    //3회실행
    var answer = exp.exec( str );
        console.log( answer[0] );
        console.log( answer[1] );
        console.log( answer[2] );
        console.log( answer[3] );



//1-4
    reg = /([ㄱ-ㅎㅏ-ㅣ가-힣A-Za-z]+)([0-9]+)/g;
    txt = '웹디자이너001,웹퍼블리셔002,프론트앤드003,백앤드004,풀스택005';

        //1회전
        result = reg.exec( txt );
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );

        //2회전
        result = reg.exec( txt );
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );

        //3회전
        result = reg.exec( txt );
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );

        //4회전
        result = reg.exec( txt );
        console.log( result[0] );
        console.log( result[1] );
        console.log( result[2] );

        //5회전
```

```
                result = reg.exec( txt );
                console.log( result[0] );
                console.log( result[1] );
                console.log( result[2] );


//1-5
        // \s Space(공백문자) 반드시 공백 1칸 있어야 함.
        //   (항목 처음은 1칸이상 인정 됨)
        // \s* Space(공백문자) 공백 있어도, 없어도 가능함.
        <!-- reg = /\s*([ㄱ-ㅎㅏ-ㅣ가-힣]+[(A-Za-z)]+)\s*([0-9]+)/g; -->
        reg = /([ㄱ-ㅎㅏ-ㅣ가-힣]+[(A-Za-z)]+)\s([0-9]+)/g;
        txt = '웹디자이너(WebDesiner) 001,웹퍼블리셔(WebPubliser) 002,프론트앤드(Frontend) 003,백앤
드(Backend) 004,풀스택(Fullstack) 005';

                for(i=0; i<txt.split(',').length; i++){
                        res = reg.exec( txt );
                        for(j=0; j<3; j++){
                                console.log( res[j] );
                        }
                }



//2. .test()메소드 : 검색 후 true, false 반환
//2-1 우편번호 zip 정규표현식 : 숫자만 \d  Digit 앞3자{3} 뒤3자{3}
        var zip = /^\d{3}-\d{3}$/g; //기준 정규표현식

                //zip.test(입력값); 문법

        //예문1]
        var str = '123-456';
        var answer = zip.test(str);
                console.log(' 우편번호 검색 : 123-456 ', answer );
                //결과: true


        //예문2]
        var str = '123456';
        var answer = zip.test(str);
                console.log(' 우편번호 검색 : 123456 ', answer );
                //결과: false : '-' 기호문자가 없어서
```

```
        //예문3]
        var str = '12-456';
        var answer = zip.test(str);
                console.log(' 우편번호 검색 : 12-456 ', answer );
                //결과: false : 앞 숫자3자인데 2자만 있어서



//2-2 우편번호 zip 정규표현식 : 숫자만 \d  Digit 앞3자{2,3} 뒤3자{2,3}
        var zip = /^\d{2,3}-\d{2,3}$/g; //기준 정규표현식 2~3자이내 제한조건

        //예문1]
        var str = '12-456';
        var answer = zip.test(str);
                console.log(' 우편번호 검색 : 12-456 ', answer );
                //결과: true


//2-3 휴대폰 전화번호 정규표현식 : 숫자만 \d  Digit 앞3자{3,4} 앞3자{3,4} 뒤4자{4}
        var hp = /^\d{3}-\d{3,4}-\d{4}$/g; //기준 정규표현식 3-4자이내 제한조건

        //예문1]
        var str = '010-7942-5305';
        var answer = hp.test(str);
                console.log(' 휴대폰번호 검색 : 010-7942-5305 ', answer );
                //결과: true


//2-4 집 전화번호 정규표현식 : 숫자만 \d  Digit 앞3자{3,4} 앞3자{3,4} 뒤4자{4}
        var tel1 = /^\d{2,3}-\d{3,4}-\d{4}$/g; //기준 정규표현식 2-3, 3-4자이내 제한조건
        var tel2 = /^\d{2,3}-\d{3,4}-\d{4}$/g; //기준 정규표현식 2-3, 3-4자이내 제한조건
        var tel3 = /^\d{2,3}-\d{3,4}-\d{4}$/g; //기준 정규표현식 2-3, 3-4자이내 제한조건

        //예문1]
        var str1 = '0233-348-6441';
        var str2 = '0700-7942-5305';
        var str3 = '0628-7942-5305';
        var answer1 = tel1.test(str1);
        var answer2 = tel2.test(str2);
        var answer3 = tel3.test(str3);
                console.log(' 전화번호 검색 : ' + str1, answer1 );
                console.log(' 전화번호 검색 : ' + str2, answer2 );
```

```
                console.log(' 전화번호 검색 : ' + str3, answer3 );
                //결과: true


//2-5 집 생년월일 정규표현식 : 숫자만 \d  Digit YYYY-MM-DD
        var birth = /^\d{4}-\d{1,2}-\d{1,2}$/g; //기준 정규표현식

        //예문1]
        var str = '1979-09-29';
        var answer = birth.test(str);
                console.log(' 생년월일 검색 : 1979-09-29 ', answer );
                //결과: true


//2-6 자동차번호 정규표현식 :  예] 21주5305
        //조건1 : 처음 2자는 숫자
        //조건2 : 중간에 한글 1자
        //조건3 : 마지막 4자는 숫자
        //모든조건 만족 /g
        var carNum     = /^\d{2}[가-힣ㄱ-ㅎㅏ-ㅣ]{1}\d{4}$/g;

        //예문1]
        var str = '21주5305';
        var answer = carNum.test(str);
                console.log(' 자동차번호 :'+str , answer );
                //결과: true



//2-7 사업자번호 정규표현식 :  예] 789-01-12345
        var buseNum    = /^\d{3}-\d{2}-\d{5}$/g;
        //예문1]
        var str = '789-01-12345';
        var answer = buseNum.test(str);
                console.log(' 사업자번호번호 :'+str , answer );
                //결과: true




//2-8 주민등록번호 정규표현식 : 예] 790929-1671023
        var persNum    = /^\d{6}-\d{7}$/g;
        //예문1]
```

```
        var str = '790929-1671023';
        var answer = persNum.test(str);
                console.log(' 주민등록번호 :'+str , answer );
                //결과: true




//3. .replace()메소드
//문법 : 문자열.replace('정규표현식', '바꿀문자열')메소드

//3-1 문자열 처음 공백제거
var reg3_1 = /^\s*/; //공백문자 제거
var str3_1 = '    Frontend Backend ';
        console.log( str3_1 ); //공백이 포함된 문자열 출력

        var result3_1 = str3_1.replace(reg3_1, '');  //문자열 앞 공백제거
                console.log( result3_1 );    //공백제거된 결과물 출력


//3-2 문자열 마지막(끝) 공백제거
var reg3_2 = /\s*$/; //공백문자 제거
var str3_2 = '    Frontend Backend       ';
        console.log( str3_2 ); //공백이 포함된 문자열 출력

        var result3_2 = str3_2.replace(reg3_2, '');  //문자열 앞 공백제거
                console.log( result3_2 );    //공백제거된 결과물 출력



//3-3 문자열 처음,마지막(끝) 공백제거
var reg3_3 = /(^\s*)|(\s*$)/g; //공백문자 제거
var str3_3 = '    Frontend     Backend       ';
        console.log( str3_3 ); //공백이 포함된 문자열 출력

        var result3_3 = str3_3.replace(reg3_3, '');  //문자열 앞 공백제거
                console.log( result3_3 );    //공백제거된 결과물 출력


//3-4 문자열 전체 공백제거
var reg3_4 = /(\s*)/g; //공백문자 제거
var str3_4 = '    Frontend     Backend       ';
```

```
        console.log( str3_4 ); //공백이 포함된 문자열 출력

        var result3_4 = str3_4.replace(reg3_4, '');  //문자열 앞 공백제거
                console.log( result3_4 );    //공백제거된 결과물 출력



//3-5 주민등록번호 마스킹 처리  . 점(dot) . 1자
//    주민번호 뒤 6자리만 마스킹처리 ******
        //예] '790929-1671023'  >> '790929-1******';
        var maskReg3_5    = /.{6}$/;  //마스킹 정규표현식  끝에서 6자
        var persNum3_5 =  '790929-1671023';

        var result3_5 = persNum3_5.replace(maskReg3_5,'******');
                console.log( result3_5 );




//3-6 사업자등록번호 마스킹 처리  . 점(dot) . 1자
//    사업자등록번호 뒤 4자리만 마스킹처리 ****
        //예] '789-01-12345'  >> '789-01-1****';
        var maskReg3_6    = /.{4}$/;  //마스킹 정규표현식  끝에서 6자
        var persNum3_6 =  '789-01-12345';

        var result3_6 = persNum3_6.replace(maskReg3_6,'****');
                console.log( result3_6 );



//3-7 문자열과 숫자가 섞여 있는 데이터를 숫자만 남기 모두 삭제
        var delReg = /\D/g;  //숫자가 아님 모든 문자열
        var txt3_7 = '웹디자이너101,웹퍼블리셔202,프론트앤드303,백앤드404,풀스택505';

        var result3_7 = txt3_7.replace(delReg, ''); //숫자를 제외한 모든 문자열 삭제
                console.log( result3_7 );


//3-8 문자열중 특정 수치를 변경
        //예] Payment : 2600000;
        //이번에 승지하여 급여가 15% 인상된다.
        //데이터를 치환하라.
```

```
        var payReg3_8 = /\d+/;
        var str3_8 =  'Payment : 2600000 원';
        var result3_8;
        /*
                function paymentFn(){
                        result3_8 = str3_8.replace(payReg3_8, 2600000*1.15);
                        console.log(result3_8);
                }

                paymentFn();
        */

        var paymentFn = str3_8.replace(payReg3_8, function(){
                return 2600000*1.15;
        });

        console.log('승진 급여 : ' + paymentFn );
```

```
//4. .match() 검색하여 맞는지(일치하는 경우) 안맞는지(일치하지않은경우) 결과 값을 배열로 반환한다.
//   /g 배열은 모두 사용
//   문법 : 문자열.match( 정규표현식 );

//4-1. 대상문자열 : 항목별 수치만 일치 검색 배열 저장
        var str4_1 = 'webdesign=2600000, frontend=2800000, backend=3000000';
        var matchReg4_1 = /[0-9]+/g;

                var result4_1 = str4_1.match( matchReg4_1 );
                        console.log('4_1 배열의 길이 : ', result4_1.length );
                        console.log('4_1 배열인수 : ', result4_1 );

                //결과 : 4_1 배열의 길이 : 3
                //결과 : 4_1 배열인수 :  (3) ["2600000", "2800000", "3000000"]
```

```
//4-2. 대상문자열 : 항목별 수치만 일치 검색 배열 저장
        var str4_2 = 'webdesign=2600000, frontend=2800000, backend=3000000';
        var matchReg4_2 = /[0-9]+/;

                var result4_2 = str4_2.match( matchReg4_2 );
                        console.log('4_2 배열의 길이 : ', result4_2.length );
                        console.log('4_2 배열인수 : ', result4_2 );

                //결과 : 4_2 배열의 길이 : 1
                //결과 : 4_2 배열인수 :  (1) ["2600000"]
                //이유 : 옵셥이 없기 때문에 /g,    Global




//4-3. 대상문자열  : 항목별 일치 검색 배열 저장
        var str4_3 = 'webdesign=2600000, frontend=2800000, backend=3000000';
        var matchReg4_3 = /[A-Za-z]+\=[0-9]+/g;
        //var matchReg4_3 =    /[A-Za-z] +          \=          [0-9]+          /g;
                                        //    /[영문대소문자]1자이상        = 등호표기    [숫자]1자이상
/옵션 전체검색대상(Global)

                var result4_3 = str4_3.match( matchReg4_3 );
                        console.log('4_3 배열의 길이 : ', result4_3.length );
                        console.log('4_3 배열인수 : ', result4_3 );

                //결과 : 4_3 배열의 길이 : 3
                //결과    :    4_3    배열인수    :    (3)    ['webdesign=2600000',    'frontend=2800000',
'backend=3000000']
                //이유 :




//4-4. 대상문자열 : 항목별 일치 검색 배열 저장
        var str4_4 = 'webdesign=2600000, frontend=2800000, backend=3000000';
        var matchReg4_4 = /\w+\=\d+/g;

                var result4_4 = str4_4.match( matchReg4_4 );
                        console.log('4_4 배열의 길이 : ', result4_4.length );
                        console.log('4_4 배열인수 /\w+\=\d+/g : ', result4_4 );
```

```
		//결과 : 4_4 배열의 길이 : 3
		//결과 : 4_4 배열인수 : (3) ['webdesign=2600000', 'frontend=2800000',
'backend=3000000']
		//이유 :



//4-5. 대상문자열 : 항목별 일치 배열 저장
	var str4_5 = 'webdesign=2600000, frontend=2800000, backend=3000000';
	var matchReg4_5 = /\w+/g;

		var result4_5 = str4_5.match( matchReg4_5 );
			console.log('4_5 배열의 길이 : ', result4_5.length );
			console.log('4_5 배열인수 /\w+/g : ', result4_5 );

		//결과 : 4_5 배열의 길이 : 3
		//결과 : 4_5 배열인수 : (3) ['webdesign=2600000', 'frontend=2800000',
'backend=3000000']
		//이유 :



//4-6. 대상문자열 : 항목별 일치 배열 저장
	var str4_6 = 'webdesign=2600000, frontend=2800000, backend=3000000';
	var matchReg4_6 = /\w+\=+/g;

		var result4_6 = str4_6.match( matchReg4_6 );
			console.log('4_6 배열의 길이 : ', result4_6.length );
			console.log('4_6 배열인수 /\w+\=+/g : ', result4_6 );

		//결과 : 4_6 배열의 길이 : 3
		//결과 : 4_6 배열인수 : (3) ['webdesign=2600000', 'frontend=2800000',
'backend=3000000']
		//이유 :


//4-7. 대상문자열 : 문자열만 일치 검색 배열 저장(=등호 수치제외)
	var str4_7 = 'webdesign=2600000, frontend=2800000, backend=3000000';
	//var matchReg4_7 = /[A-Z]+/gi;
	var matchReg4_7 = /[a-zA-Z]+/g;
```

```
        var result4_7 = str4_7.match( matchReg4_7 );
            console.log('4_7 배열의 길이 : ', result4_7.length );
            console.log('4_7 배열인수 /[a-zA-Z]+/g : ', result4_7 );

        //결과 : 4_7 배열의 길이 : 3
        //결과 : 4_7 배열인수 : (3) ['webdesign', 'frontend', 'backend']



//4-8. 대상문자열 : WebSite 일치 검색 배열 저장
    var str4_8 = 'Email=moonseonjong@naver.com, WebSite=http://www.moonjong.co.kr, HP=010-7942-5305';
                                // /WebSite = http:// www . moonjong . co . kr, HP=010-7942-5305';
//  var matchReg4_8 = /[a-zA-Z]+ \= http:\/\/ \w+ \. \w+ \. \w+ \. \w+ /g;
    var matchReg4_8 = /[a-zA-Z]+\=http:\/\/\w+\.\w+\.\w+\.\w+/g;


        var result4_8 = str4_8.match( matchReg4_8 );
            console.log('4_8 배열의 길이 : ', result4_8.length );
            console.log('4_8 배열인수 /[a-zA-Z]+\=http:\/\/\w+\.\w+\.\w+\.\w+/g : ', result4_8 );


            //결과 : 4_8 배열의 길이 : 1
            //결과 : 4_8 배열인수 : (3) ['WebSite=http://www.moonjong.co.kr']



//4-9. 대상문자열 : Email 일치 검색 배열 저장
    var str4_9 = 'Email=moonseonjong@naver.com, WebSite=http://www.moonjong.co.kr, HP=010-7942-5305';
                                // /Email = moonseonjong @ naver . com
//  var matchReg4_9 = /[a-zA-Z]+ \= \w+ \@ \w+ \. \w+ /g;
    var matchReg4_9 = /[a-zA-Z]+\=\w+\@\w+\.\w+/g;


        var result4_9 = str4_9.match( matchReg4_9 );
            console.log('4_9 배열의 길이 : ', result4_9.length );
```

```
                    console.log('4_9 배열인수  /[a-zA-Z]+\=\w+\@\w+\.\w+/g: ', result4_9 );


                    //결과 : 4_9 배열의 길이 : 1
                    //결과 : 4_9 배열인수 : ["Email=moonseonjong@naver.com"]









//4-10. 대상문자열 : HP 일치하는 정보 검색 배열 저장
        var  str4_10  =  'Email=moonseonjong@naver.com,     WebSite=http://www.moonjong.co.kr,
HP=010-7942-5305';
                                    //   / HP         = 010     -  7942       -   5305'
        //var matchReg4_10 = /[a-zA-Z]+ \= \d{3} \-  \d{3,4}  \-   \d{4} /g;

        var matchReg4_10 = /[a-zA-Z]+\=\d{3}\-\d{3,4}\-\d{4}/g;

                var result4_10 = str4_10.match( matchReg4_10 );
                    console.log('4_10 배열의 길이 : ', result4_10.length );
                    console.log('4_10   배열인수   /[a-zA-Z]+\=\d{3}\-\d{3,4}\-\d{4}/g      :   ',
result4_10 );


                    //결과 : 4_10 배열의 길이 : 1
                    //결과 : 4_10 배열인수 :  ["HP=010-7942-5305"]




//4-11. 대상문자열 : 다양한 형태의 이메일 일치하는 정보 검색 배열 저장
        //      형식1]  moonseonjong@naver.com
        //      형식2]  moon-seon-jong.dana@yahoo.co.kr
        //      형식3]  moon_seon_jong@yahoo.co.kr
        //      형식4]  moonjong29@yahoo.co.kr


        var  str4_11  =  'Email=moonseonjong@naver.com,     Email=moon_seon_jong@yahoo.co.kr,
```

```
Email=moon-seon-jong.dana.mo@yahoo.co.kr';

                                        // 이메일 형식1]
                                        // moonseonjong @ naver . com

        //var matchReg4_11 = /[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+/g;


                                        // 이메일 형식2 형식3 형식4]
                                        // @앞부분 [- _ . ]추가 변형된 형식
                                        // @뒷부분 . 추가 변형된 형식(_-추가 가능성 있음)

                                        // moon - seon - jong . dana @ yahoo . co . kr
                                        // moon _ seon _ jong @ yahoo . co . kr

                                        // moonjong29@yahoo.co.kr

        //var matchReg4_11 = /[a-zA-Z0-9]+     ([_-.]?[A-Za-z0-9])*          @  [a-zA-Z0-9]+
([_-.]?[0-9a-zA-Z])*  \.[a-zA-Z0-9]+/g;
        //var                                   matchReg4_11                                    =
/[a-zA-Z0-9]+([_-.]?[A-Za-z0-9])*@[a-zA-Z0-9]+([_-.]?[0-9a-zA-Z])*\.[a-zA-Z0-9]+/g;


        var                                   matchReg4_11                                    =
/[a-zA-Z0-9]+([\_\-\.]?[A-Za-z0-9])*@[a-zA-Z0-9]+([\_\-\.]?[0-9a-zA-Z])*\.[a-zA-Z0-9]+/g;
        //var matchReg4_11 = /[a-zA-Z0-9]+@[a-zA-Z0-9]+[a-zA-Z0-9]+/g;

                var result4_11 = str4_11.match( matchReg4_11 );
                        console.log('4_11 배열의 길이 : ', result4_11.length );
                        console.log('4_11                                                                배열인수
/[a-zA-Z0-9]+([\_\-\.]?[A-Za-z0-9])*@[a-zA-Z0-9]+([\_\-\.]?[0-9a-zA-Z])*\.[a-zA-Z0-9]{2,10}/g  :  ',
result4_11 );


                        //결과 : 4_11 배열의 길이 :
                        //결과 : 4_11 배열인수 :




//5.search();
//5-1. 휴대폰 번호 검색
```

```
//  정보 검색 포지션(Position) 위치를(index) 수치로 나타낸다.
//  첫번째 위치는 0 두번째 우치는 1 .....  Index number
//  문법 : 문자열.search( 정규표현식 );
//  대상 문자열
var str5_1 = 'HP:010-7942-5305, TEL:070-7942-5305';
var hpReg5_1 = /\d{3}\-\d{3,4}\-\d{4}/g; //휴대폰번호 검색 정규표현식

        var result5_1 = str5_1.search( hpReg5_1 );
                        console.log('5_1 검색 RegExp ' + hpReg5_1 +  ' : ', result5_1 );
                //결과 : 5_1 포지션 : 3
                // 010-7942-5305 검색 처음 위치의 포진션 인덱스 값 4번째 위치




//5-2. 휴대폰 번호 검색
//  정보 검색 포지션(Position) 위치를(index) 수치로 나타낸다.
//  첫번째 위치는 0 두번째 우치는 1 .....  Index number
//  문법 : 문자열.search( 정규표현식 );
//  대상 문자열
var str5_2 = 'HP:010-7942-5305, TEL:070-7942-5305';
var hpReg5_2 = /070\-\d{3,4}\-\d{4}/g; //휴대폰번호 검색 정규표현식

        var result5_2 = str5_2.search( hpReg5_2 );
                        console.log('5_2 검색 RegExp ' + hpReg5_2 +  ' : ', result5_2 );
                //결과 : 5_2 포지션 : 3
                // 010-7942-5305 검색 처음 위치


//5-3. 휴대폰 번호 검색
//  정보 검색 포지션(Position) 위치를(index) 수치로 나타낸다.
//  첫번째 위치는 0 두번째 우치는 1 .....  Index number
//  문법 : 문자열.search( 정규표현식 );
//  대상 문자열
//var str5_3 = 'HP:017-7942-5305, TEL:070-7942-5305';
//var str5_3 = 'HP:016-7942-5305, TEL:070-7942-5305';
//var str5_3 = 'HP:011-7942-5305, TEL:070-7942-5305';
var str5_3 = 'HP:017-7942-5305, TEL:070-7942-5305';
var hpReg5_3 = /01[0|1|2|3|4|5|6|7|8|9]+\-\d{3,4}\-\d{4}/g; //휴대폰번호 검색 정규표현식
        //전화번호 앞자리 3자리인데 010- ... 011- ... 012- ... 019-
        //모두 검색 대상이 된다
```

```
        var result5_3 = str5_3.search( hpReg5_3 );
                        console.log('5_3 검색 RegExp /01[0|1|2|3|4|5|6|7|8|9]+\-\d{3,4}\-\d{4}/g : ' ,
result5_3 );
                //결과 : 5_3 포지션 : 3
                // 010-7942-5305 검색 처음 위치



////////////////////////////////////////////////////
//응용 replace(); 치환
//1. 콤머 표기하기
var str1 = '123';
var reg1 = /\d+/g;
var result1 = str1.replace(reg1, ',');
console.log( result1 );
//결과 ,



//2. 콤머 표기하기
var str2 = '123';
var reg2 = /\B/g;
var result2 = str2.replace(reg2, ',');
console.log( result2 );
//결과 1,23;




//3. 콤머 표기하기
var str3 = '123';
          '1 첫글자다음 숫자 3자 앞에만 콤머 찍기 23'
          //첫글자 다음 바로 연이어서 ?=
          //숫자 3자 \d{3}가 있는 경우
          //메타문자 ?=\d{3}
var reg3 = /\B(?=\d{3})/g;
var result3 = str3.replace(reg3, ',');
console.log( result3 );
//결과 123;
```

```
//수치가 - 문자열 스트링변환() toString()

//4. 콤머 표기하기
var str4 = 1234;
console.log( str4.toString() );
          '1 첫글자다음 숫자 3자 앞에만 콤머 찍기 23'
          //첫글자 다음 바로 연이어서 ?=
          //숫자 3자 \d{3}가 있는 경우
          //메타문자 ?=\d{3}


var reg4 = /\B(?=\d{3})/g;
var result4 = str4.toString().replace(reg4, ',');
console.log( result4 );
//결과 1,234




//5. 콤머 표기하기
var str5 = '1234567890';
          '1 첫글자다음 숫자 3자 앞에만 콤머 찍기 23'
          //첫글자 다음 바로 연이어서 ?=
          //숫자 3자 \d{3}가 있는 경우
          //메타문자 ?=\d{3}
var reg5 = /\B(?=(\d{3})+(?!(\d)))/g;
var result5 = str5.replace(reg5, ',');
console.log( result5 );
//결과 1,234567890
//맨 앞에만 콤머 표기 문제



//6. 콤머 표기하기
var str6 = '1234567890'; // 스트링(문자열) string
          '1 첫글자다음 숫자 3자 앞에만 콤머 찍기 23'
          //첫글자 다음 바로 연이어서 ?=
          //숫자 3자 \d{3}가 있는 경우
          //메타문자 (?=\d{3})
          //뒤 숫자가 아닌 문자를 찾아서 다음 숫자 3자리 마다 콤머를 찍는다. 의미
          //!=(부정) 메타문자에는 !(부정) \d
          // (?!\d)  위에 숫자가 아니면
var reg6 = /\B           ( ?=        (\d{3})+     (?!(\d))     )/g;
```

```
            //단어가아닌 첫글자   (  연이어서나온    숫자3자  필수      숫자가아닌것        /Global  검색
    )



var reg6 = /\B(?=(\d{3})+(?!(\d)))/g;
var result6 = str6.replace(reg6, ',');
console.log( result6 );
//결과 2,431,233,342,343,297,890
//뒤 숫자가 아닌 문자를 찾아서 다음 숫자 3자리 마다 콤머를 찍는다. 의미




</script>
```