Javascript

정규 표현식

1 정규 표현식 객체

- **정규 표현식 객체**
  - 정규 표현식 객체는 자바스크립트의 기본 내장 객체 중 하나
  - 정규 표현식 객체는 두 가지 방법으로 생성

```
<script>
    var regExp1 = new RegExp('text');
    var regExp2 = /text/;
</script>
```

1 정규 표현식 객체

- **정규 표현식 객체**
  - 정규 표현식을 사용한 예제

메서드 이름	설명
test( )	정규 표현식과 일치하는 문자열이 있으면 true를, 아니면 false를 리턴합니다.
exec( )	정규 표현식과 일치하는 문자열을 리턴합니다.

```
<script>
    // 변수를 선언합니다.
    var regExp = /script/;
    var string = 'Javascript jQuery Ajax';

    // 메서드를 사용합니다.
    var output = regExp.test(string);

    // 출력합니다.
    alert(output);
</script>
```

1 정규 표현식 객체

- **정규 표현식 객체**
  - 정규 표현식의 메서드를 곧바로 사용하는 것보다는 문자열 객체의 메서드와 함께 사용하는 것이 일반적

메서드 이름	설명
match(regExp)	정규 표현식과 일치하는 부분을 리턴합니다.
replace(regExp, replacement)	정규 표현식과 일치하는 부분을 새로운 문자열로 바꿉니다.
search(regExp)	정규 표현식과 일치하는 부분의 위치를 리턴합니다.
split(regExp)	정규 표현식을 기준으로 문자열을 잘라 배열을 리턴합니다.

```
<script>
    // 변수를 선언합니다.
    var regExp = /script/;
    var string = 'Javascript jQuery Ajax';

    // 메서드를 사용합니다.
    var output = string.split(regExp);

    // 출력합니다.
    alert(output);
</script>
```

2 대체 문자

- **대체 문자**
  - 정규 표현식을 사용하면 문자열 객체의 replace ( ) 메서드를 사용할 때 대체 문자를 사용할 수 있음

정규 표현식 기호	설명
$&	일치하는 문자열
$`	일치하는 부분의 앞부분 문자열
$'	일치하는 부분의 뒷부분 문자열
$1, $2, $3	그룹

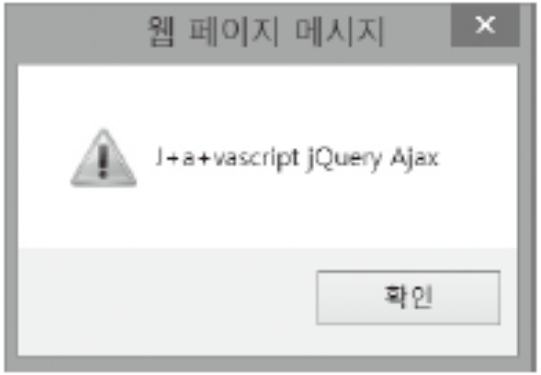
2 대체 문자

- **대체 문자**
  - 문자열 객체의 replace ( ) 메서드로 정규 표현식에 일치하는 문자열을 '+$&+'로 변경

```
<script>
    // 변수를 선언합니다.
    var regExp = /a/;
    var string = 'Javascript jQuery Ajax';

    // 메서드를 사용합니다.
    var output = string.replace(regExp, '+$&+');

    // 출력합니다.
    alert(output);
</script>
```



2 대체 문자

- **대체 문자**
  - 문자열 객체의 replace ( ) 메서드의 매개변수에는 이렇게 함수를 넣을 수 있음
  - 매개변수로 넣은 함수의 매개변수에는 정규 표현식에 대응하는 문자열이 입력되며 리턴하는 문자열로 대체

```
// 메서드를 사용합니다.
var output = string.replace(regExp, function (value) {
    return '+' + value + '+';
});
```

3 플래그 문자

- **플래그 문자**
  - 플래그 문자 위치에 들어가는 플래그 문자의 순서는 어떻게 구성되어도 상관없음

정규 표현식 기호	설명
g	전역 비교를 수행합니다.
i	대소문자를 가리지 않고 비교합니다.
m	여러 줄의 검사를 수행합니다.

```
var regExp = /Expression/im
var regExp = new regExp('Expression', 'im');
```

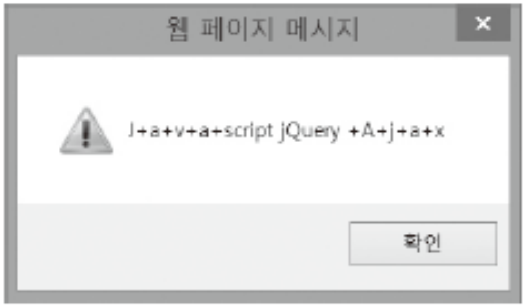
3 플래그 문자

- 플래그 문자

```
<script>
    // 변수를 선언합니다.
    var regExp = /a/ig;
    var string = 'Javascript jQuery Ajax';

    // 메서드를 사용합니다.
    var output = string.replace(regExp, '+$&+');

    // 출력합니다.
    alert(output);
</script>
```



4 앵커 문자

- 앵커 문자

정규 표현식 기호	설명
^ABC	맨 앞 문자가 ABC
ABC$	맨 뒤 문자가 ABC

```
<script>
    // 변수를 선언합니다.
    var regExp = /^j/ig;
    var string = 'Javascript\njQuery\nAjax';

    // 메서드를 사용합니다.
    var output = string.replace(regExp, '+$&+');

    // 출력합니다.
    alert(output);
</script>
```

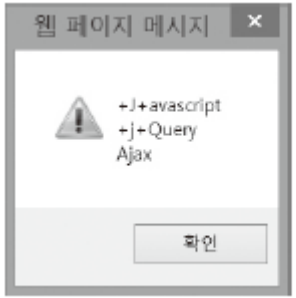
4 앵커 문자

- 앵커 문자
  - 플래그 문자 m은 그림 B-5처럼 문자열이 여러 줄을 형성할 때, 각각의 줄을 개별적인 문자열로 인지하고 검사할 수 있게 해주는 플래그 문자

```
<script>
    // 변수를 선언합니다.
    var regExp = /^j/igm;
    var string = 'Javascript\njQuery\nAjax';

    // 메서드를 사용합니다.
    var output = string.replace(regExp, '+$&+');

    // 출력합니다.
    alert(output);
</script>
```



5 메타 문자

- 메타 문자

기호	설명
.	아무 글자
[abc]	괄호 안의 글자
[^abc]	괄호 안의 글자 제외
[a-z]	알파벳 a부터 z까지
[A-Z]	알파벳 A부터 Z까지
[0-9]	숫자 0부터 9까지

```
<script>
    // 변수를 선언합니다.
    var regExp = /[aj]/ig;
    var string = 'Javascript jQuery Ajax';

    // 메서드를 사용합니다.
    var output = string.replace(regExp, '+$&+');

    // 출력합니다.
    alert(output);
</script>
```

5 메타 문자

- 메타 문자

기호	설명
\d	숫자
\w	아무 단어(숫자 포함)
\s	공백 문자(탭, 띄어쓰기, 개행)
\D	숫자 아님
\W	아무 단어 아님
\S	공백 문자 아님

6 수량 문자

- 수량 문자

정규 표현식 기호	설명
a+	a가 적어도 1개 이상
a*	a가 0개 또는 여러 개
a?	a가 0개 또는 1개
a{5}	a가 5개
a{2,5}	a가 2개~5개
a{2,}	a가 2개 이상
a{,2}	a가 2개 이하

7 선택 문자

- 선택 문자

정규 표현식 기호	설명
(abc|def)	abc 또는 def를 선택합니다.

```
<script>
    // 변수를 선언합니다.
    var string = prompt('소문자 또는 숫자로만 구성된 단어를 입력하세요.', '단어');
    var regExp = /([0-9]|[a-z])/g;

    // 출력합니다.
    if (string.replace(regExp, '').length == 0) {
        alert('감사합니다.');
    } else {
        alert('ㅇㅁㅇ');
    }
</script>
```

6 정규 표현식 사용 예제

- **정규 표현식 사용 예제**
  - 한글 이름을 확인하는 정규 표현식
    - 모든 한글이 빈 문자열로 바뀌었다면 문자열 replacedString의 length 속성은 0이 되므로 한글로만 구성돼 있는지 확인

```
<script>
    function isKoreanName(string) {
        // 변수를 선언합니다.
        var regExp = /([가-힣])/g;

        // 메서드를 사용합니다.
        var replacedString = string.replace(regExp, '');

        // 확인합니다.
        if (replacedString.length == 0) {
            return true;
        } else {
            return false;
        }
    }
```

6 정규 표현식 사용 예제

- **정규 표현식 사용 예제**
  - 이메일을 구분하는 방법

```
<script>
    function isEmail(string) {
        // 변수를 선언합니다.
        var regExp = /\w+@\w+\.\w+/;
        // 확인합니다.
        return regExp.test(string);
    }

    alert(isEmail('rintianttа@naver.com'));
</script>
```

정규식 패턴

- **정규 표현식 문법**
  - 정규 표현식을 배우기 위해 기본으로 알아야 하는 개념은 메타문자(meta-characters) 이다. 메타문자는 문자를 설명하기 위한 문자로, 문자의 구성을 설명하기 위해 원래의 의미가 아니라 다른 의미로 쓰이는 문자를 뜻한다.
  - 기본 메타문자 []
    - 먼저 대괄호 [ ]는 [ ] 블록 사이의 문자와 매칭. [ ]에는 or의 의미.
      예를 들어, [abc]는 어떤 텍스트에 a 또는 b 또는 c라는 텍스트가 있는지 확인
  - 반복 관련 메타문자 -, +, *, ?, { }
    - + : 해당 글자가 1개 이상 출현
    - {} : 출현 횟수를 조정해야 할 때 사용하는 메타문자는 중괄호
    - * : 해당 글자가 0번부터 무한대까지 반복
    - () : 묶음을 표시
      [.]는 일반적인 마침표를 뜻하고 (.)는 줄 바꿈 기호를 제외한 전체 문자를 뜻한다.
    - 메타문자 |나 ^은 or와 not의 의미, 정규 표현식의 처음과 끝에는 메타문자 ^과 $를 붙인다.

```
. ^ $ * + ? {} [] \ | ()
```

정규식 패턴

- **정규 표현식 문법**
  - 전화번호 찾기

```
[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]
```

```
[0-9]+-[0-9]+-[0-9]+
```

```
[0-9]{3}-[0-9]{3,4}-[0-9]{4}
```

정규식 패턴

- ^ (caret)
  - 라인의 처음이나 문자열의 처음을 표시
  - 예 : ^aa (문자열의 처음에 aa를 포함하면 참, 그렇지 않으면 거짓)
- $ (dollar)
  - 라인의 끝이나 문자열의 끝을 표시
  - 예 : aaa$ (문자열의 끝에 aaa를 포함하면 참, 그렇지 않으면 거짓)
- . (period)
  - 임의의 한 문자를 표시
  - 예 : ^a.c (문자열의 처음에 abc, adc, aZc 등은 참, aa 는 거짓)
  - 예 : a..b$ (문자열의 끝에 aaab, abbb, azzb 등을 포함하면 참)
- [] (bracket)
  - 문자의 집합이나 범위를 나타냄, 두 문자 사이의 범위는 "-" 사용.
  - []내에서 "^"이 선행되면 not을 나타냄
  - 예 : [abc] (a, b, c 중 어떤 문자, "[a-c]."과 동일)
  - 예 : [Yy] (Y 또는 y)
  - 예 : [A-Za-z0-9] (모든 알파벳과 숫자)
  - 예 : [-A-Z]. ("-"(hyphen)과 모든 대문자)
  - 예 : [^a-z] (소문자 이외의 문자)
  - 예 : [^0-9] (숫자 이외의 문자)

정규식 패턴

- {} (brace)

  {} 내의 숫자는 직전의 선행 문자가 나타나는 횟수,범위를 나타냄

  예 : a{3} ('a'의 3번만 반복인 aaa만 해당됨)

  예 : a{3,} ('a'가 3번 이상 반복인 aaa, aaaa, … 등을 나타냄)

  예 : a{3,5} (aaa, aaaa, aaaaa 만 해당됨)

  예 : ab{2,3} (abb와 abbb 만 해당됨)

  예 : [0-9]{2} (두 자리 숫자)

  예 : doc[7-9]{2} (doc77, doc87, doc97 등이 해당)

  예 : [^Zz]{3} (Z와 z를 포함하지 않는 5개의 문자열, abc, ttt 등)

  예 : .{3,4}er ('er'앞에 세 개 또는 네 개의 문자를 포함하는 문자열이므로 Peter, mother 등이 해당)

- * (asterisk)

  "*" 직전의 선행 문자가 0번 또는 여러 번 나타나는 문자열

  예 : ab*c ('b'를 0번 또는 여러 번 포함하므로 ac, abc, abbbc 등)

  예 : * (선행 문자가 없는 경우이므로 임의의 문자열 및 공백 문자열)

  예 : .* (선행 문자가 "."이므로 하나이상의 문자를 포함하는 문자열)

  예 : ab* ('b'를 0번 또는 여러 번 포함하므로 a, accc, abb 등)

  예 : a* ('a'를 0번 또는 여러 번 포함하므로 k, kd, a, aa, abb 등)

  예 : doc[7-9]* (doc7, doc777, doc778989, doc 등이 해당)

  예 : [A-Z].* (대문자로만 이루어진 문자열)

  예 : like.* (직전의 선행 문자가 '.'이므로 like에 0 또는 하나 이상의 문자가 추가된 문자열이 됨, like, likely, liker, likelihood 등)

정규식 패턴

- + (Plus Sign)

  "+" 직전의 선행 문자가 1번 이상 나타나는 문자열

  예 : ab+c ('b'를 1번 또는 여러 번 포함하므로 abc, abcd, abbc 등)

  예 : ab+ ('b'를 1번 또는 여러 번 포함하므로 ab, abcc, abb 등)

  예 : [A-Z]+ (대문자로만 이루어진 문자열)

  예 : like.+ (직전의 선행 문자가 '.'이므로 like에 하나 이상의 문자가 추가된 문자열이 됨, likely, liker, likelihood 등, 그러나 like는 해당 안됨)

- ? (question mark)

  "?" 직전의 선행 문자가 0번 또는 1번 나타나는 문자열

  예 : ab?c ('b'를 0번 또는 1번 포함하므로 abc, abcd 만 해당됨)

- () (parenthesis)

  ()는 정규식내에서 패턴을 그룹화 할 때 사용

- | (bar)

  or를 나타냄

  예 : a|b|c (a, b, c 중 하나, 즉 [a-c]와 동일함)

  예 : yes|Yes (yes나 Yes 중 하나, [yY]es와 동일함)

  예 : korea|japan|chinese (korea, japan, chinese 중 하나)

- ₩ (backslash)

  위에서 사용된 특수 문자들을 정규식 내에서 문자로 취급하고 싶을 때 '₩'를 선행시켜서 사용하면 됨

  예 : filename₩.ext ("filename.ext"를 나타냄)

  예 : [₩?₩[₩₩₩]] ('?', '[', '₩', ']' 중 하나)

정규 표현식 예제

구분	표현식	설명
한글만 입력가능	/^[가-힣]+$/	
숫자만 입력가능	/^[0-9]*$/	
영문만 입력가능	/^[a-zA-Z]+$/	
한글+ 영문만 입력가능	/^[가-힣a-zA-Z]+$/	
아이디(영문+숫자)	/^[a-zA-Z0-9]{4,12}$/	4글자 이상, 12자 미만, 영어 대소문자, 숫자, -, _ 사용 가능
비밀번호(영문/숫자+특문)	/^(?=.*[a-zA-Z])((?=.*₩d)|(?=.*₩W)).{6,20}$/	6글자 이상 20자 미만, 최소 1개의 숫자 혹은 특수 문자 포함
이메일 주소	/^[a-z0-9_+.-]+@([a-z0-9-]+₩.)+[a-z0-9]{2,4}$/	
전화번호	/^₩d{2,3}-₩d{3,4}-₩d{4}$/	
핸드폰입력	/^01([0|1|6|7|8|9]?)-?([0-9]{3,4})-?([0-9]{4})$/	
url입력	/^(file|gopher|news|nntp|telnet|https?|ftps?|sftp):₩/₩/([a-z0-9-]+₩.)+[a-z0-9]{2,4}.*$/	

- **정규 표현식 문법 연습**
  - 정규 표현식 연습장 웹 사이트(http://www.regexr.com) 활용