자바스크립트 정규표현식

우선, 정규표현식은 두 가지 단계(`컴파일` , `실행`)로 이루어진다. 먼저 `컴파일` 단계에서 어떠한 패턴을 찾아내고, `실행` 단계에서는 찾아낸 대상에 대한 어떠한 작업을 한다.

컴파일

컴파일은 검출하고자 하는 패턴을 만드는 일이다. 우선 정규표현식 객체를 만들어야 한다. 객체를 만드는 방법은 두가지가 있다. a라는 텍스트를 찾아내는 정규표현식을 만들어보자.

```
var pattern = /a/;

var pattern = new RegExp('a');
```

위 코드는 정규표현식 객체라는 것을 pattern이라는 변수에 할당하는 작업을 하는 동일한 코드이다. 하지만 두 코드는 약간의 차이점이 있다. 이 차이점은 링크를 통해 확인할 수 있다.

정규표현식 리터럴

```
var pattern =/a/;
```

위와 같은 코드를 정규표현식 리터럴이라고 부른다.

정규표현식 객체 생성자

```
var pattern = new RegExp('a');
```

위와 같은 코드를 정규표현식 객체생성자라고 부른다.

정규표현식을 컴파일해서 객체를 만들었다면 이제 문자열에서 원하는 문자를 찾아내야 한다.

RegExp.exec()

`.exec()` 라는 메소드는 `패턴.exec(대상)` 과 같은 형태로 쓰이는 데, 지정된 '패턴'이 '대상'에 있는지 확인 후 그 결과를 배열로 리턴하는 함수이다.

```
//var pattern =/a/;
console.log(pattern.exec('abcdef')); //["a"]
//var pattern =/a./;
console.log(pattern.exec('abcdef')); //["ab"]
//var pattern =/a../;
console.log(pattern.exec('abcdef')); //["abc"]
```

실행결과는 문자열 a를 값으로 하는 배열을 리턴한다. 만약 값이 없을 때는 null을 리턴한다. 패턴에서 `.` 은 문자하나를 의미한다. 즉, 위와 같이 패턴들을 정의하고 그 결과를 보면 `.` 하나당 패턴검사에서 리턴되는 문자의 갯수가 하나씩 늘어나는 것을 확인할 수 있다.

RegExp.test()

test는 인자 안에 패턴에 해당되는 문자열이 있으면 true, 없으면 false를 리턴한다.

```
//var pattern =/a/;
console.log(pattern.test('abcdef')); //true
console.log(pattern.test('bcdefg')); //false
```

문자열에서도 패턴검색을 할 수 있다.

String.match()

RegExp.exec()와 비슷하다.

```
console.log('abcdef'.match(pattern)); // ["a"]
console.log('bcdefg'.match(pattern)); // null
```

String.replace()

문자열에서 패턴을 검색하여 이를 변경한 후에 변경된 값을 리턴한다.

```
console.log('abcdef'.replace(pattern, 'A')); // abcdef
```

옵션

정규표현식 패턴을 만들 때 옵션을 설정할 수 있다. 옵션에 따라서 검출되는 데이터가 달라진다.

i

i를 붙이면 대소문자를 구분하지 않느다.

```
var xi = /a/;
console.log("Abcde".match(xi)); // null
var oi = /a/i;
console.log("Abcde".match(oi)); // ["A"];
```

g

g를 붙이면 검색된 모든 결과를 리턴한다.

```
var xg = /a/;
console.log("abcdea".match(xg)); //["a"]
var og = /a/g;
console.log("abcdea".match(og)); //["a", "a"]
```

ig

옵션을 합쳐서 사용할 수도 있다.

```
var ig = /a/ig;
console.log("AabcdAa".match(ig)); // ["A", "a", "A", "a"]
```

캡쳐

```
var pattern = /(\w+)\s(\w+)/;
var str = "coding everybody";
var result = str.replace(pattern, "$2, $1");
console.log(result); // everybody, coding
```

> 캡쳐란, 정규표현식에서 그룹을 지정하고 그 그룹을 가져와서 사용할 수 있는 개념을 의미한다. 여기서 `그룹` 은 위 `pattern` 이란 정규표현식 객체에서 `()` 로 감싸여진 `(\w+)` 을 의미한다.
>
> - `\w` 는 하나의 `문자` 를 의미. 여기서 문자란? A~Z, a~z, 0~9를 의미.
> - +는 수량자 앞에 있는 문자가 하나 이상일 때 유효한 패턴이 됨.
> - `\s` 는 스페이스(공백)을 의미.
> - `$` 는 `그룹` 을 의미하는데 `$1` , `$2` 는 각각 첫번째 그룹, 두번째 그룹을 의미한다.

따라서 위 코드의 `result` 는 정규표현식에 따라 치환된 결과인 `everybody, coding` 이 된다.

치환

아래 코드는 `content` 중의 URL을 링크 html 태그로 교체한다.

```
var urlPattern = /\b(?:https?):\/\/[a-z0-9-+&@#\/%?=~_|!:,.;]*/gim;
var content = '생활코딩 : http://opentutorials.org/course/1 입니다. 네이버 :
http://naver.com 입니다. ';
var result = content.replace(urlPattern, function(url){
    return '<a href="'+url+'">'+url+'</a>';
});
console.log(result);
```

결과는 아래와 같이 `content` 의 url들을 link가 있는 html태그로 교체하였다.

> 생활코딩 : http://opentutorials.org/course/1 입니다. 네이버 : http://naver.com 입니다.

참고

아래는 정규표현식을 공부하는데 도움이 되는 사이트이다.

- 생활코딩 정규표현식 수업
- 정규표현식을 시각화
- 정규표현식 빌더