JSON (JavaScript Object Notation)

[1] 개요

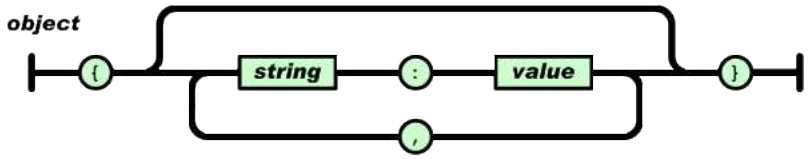
JSON(제이슨, JavaScript Object Notation)은, 인터넷에서 자료를 주고받을 때 그 자료를 표현하는 방법이다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다.

JSON (JavaScript Object Notation)은 경량의 DATA-교환 형식으로, 사람이 읽고 쓰기에 용이하며, 기계가 분석하고 생성함에도 용이하다. JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999의 일부에 토대를 두고 있으며 JSON은 완벽하게 언어로 부터 독립적이지만 C-family 언어 - C, C++, C#, Java, JavaScript, Perl, Python 그외 다수의 프로그래머들에게 친숙한 관습을 사용하는 텍스트 형식이다. 이러한 속성들이 JSON을 이상적인 DATA-교환 언어로 만들고 있다.

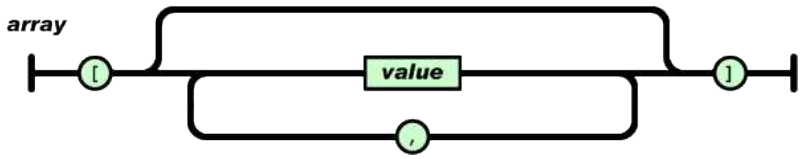
JSON은 두개의 구조를 기본으로 두고 있다.

- name/value 형태의 쌍으로 collection 타입. 다양한 언어들에서, 이는 object, record, struct(구조체), dictionary, hash table, 키가 있는 list, 또는 연상배열로서 실현 되었다.
- 값들의 순서화된 리스트. 대부분의 언어들에서, 이는 array, vector, list, 또는 sequence로서 실현 되었다.

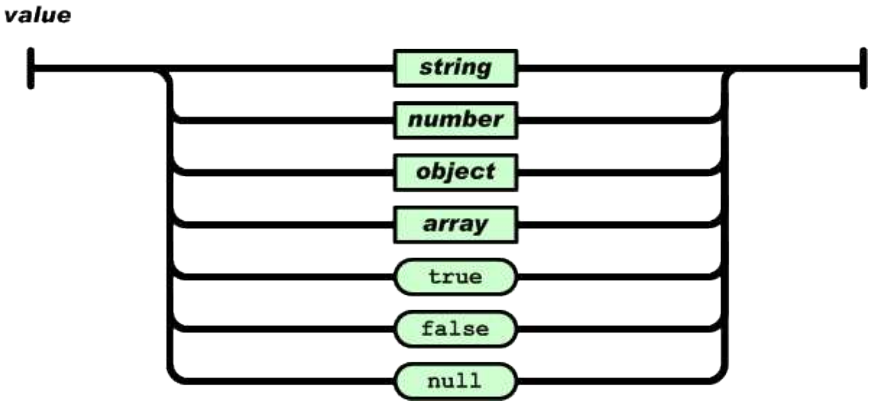
object는 name/value 쌍들의 비순서화된 SET이다. object는 { (좌 중괄호)로 시작하고 } (우 중괄호)로 끝내어 표현한다. 각 name 뒤에 : (colon)을 붙이고 , (comma)로 name/value 쌍들 간을 구분한다.



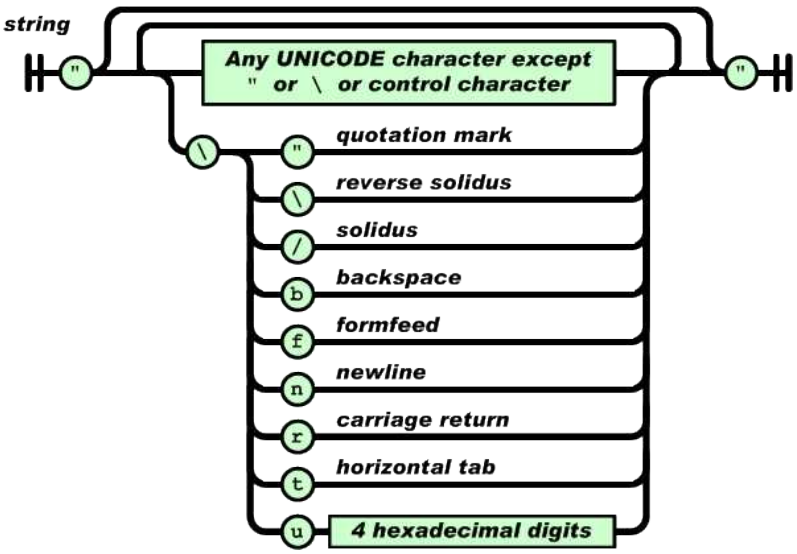
array은 값들의 순서화된 collection 이다. array는 [(left bracket)로 시작해서] (right bracket)로 끝내어 표현한다. , (comma)로 array의 값들을 구분한다.



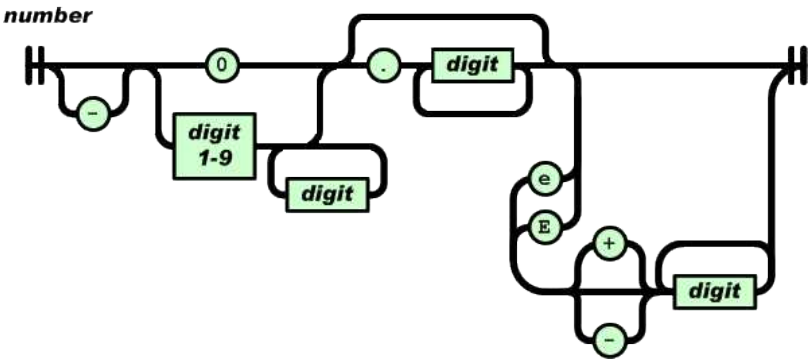
value는 큰따옴표안에 string, number ,true ,false , null, object ,array이 올수 있다. 이러한 구조들을 포함한다.



string은 큰따옴표안에 둘러 싸인 zero 이상 Unicode 문자들의 조합이며, 쌍다옴표안에 감싸지며,backslash escape가 적용된다. 하나의 문자(character)도 하나의 문자열(character string)로서 표현된다. string은 C 또는 Java 문자열 처럼 매우 많이 비슷하다.



number는 8진수와 16진수 형식을 사용하지 않는것을 제외하면 C와 Java number 처럼 매우 많이 비슷하다.



- JSON 공식 홈페이지 (http://www.JSON.org)
- 한국어페이지 (http://www.json.org/json-ko.html)

[2] JSON의 기본

기본적인 형태는 아래와 같이 사용이 가능하다.

1) 프로퍼티 <-> 값
```
var obj = {
   "프로퍼티 이름" : "값",
}
```

2) 메소드
```
var obj = {
   "메소드 이름" : function() {alert('This is method')}
}
```

3) 메소드(인수)
```
var obj = {
    "메소드 이름" : function(인수) {alert('This is method')}
}
```

이것만으로 오브젝트 obj를 만드는 것이 가능하여, obj.프로퍼티이름 으로 값을 얻어 낼 수 있어, obj.메소드이름() 으로 "This is method"라는 대화창을 표시한다.

■ 오브젝트

아래와 같이 myJSONObject를 만들어보고 이를 Javascript를 이용해 확인해보자.

```
<script language="JavaScript">
<!--
  var myJSONObject = {
    "test" : "hello"
  }
//-->
</script>
<form>
  <input type = "button" onclick="alert(typeof myJSONObject)" value="click">
</form>
```

■ 프로퍼티

```
<script language = "JavaScript">
<!--
  var myJSONObject2 = {
    "test": "hello"
  }
//-->
</script>
<form>
  <input type = "button" onclick = "alert( myJSONObject2 )" value = "click">
  <input type = "button" onclick = "alert( myJSONObject2.test )" value = "click">
  <input type = "button" onclick = "myJSONObject2.test = 'new test' ;
alert(myJSONObject2.test )" value = "click">
</form>
```

(1) alert(myJSONObject2)는 myJSONObject2가 object임을 보여준다.

(2) alert(myJSONObject2.test)는 myJSONObject2 오브젝트에서 test프로퍼티의 값 "hello"를 가져온다.

(3) myJSONObject2.test = 'new test' ;에서 test 프로퍼티의 값을 "new test"로 변경한다.

alert(myJSONObject2.test)를 다시 수행하게되면 "hello"의 값이 "new test"로 변경되었음을 확인할 수 있다.

위의 예에서는 프로퍼티가 한개인 경우를 테스트하여 보았다. 그렇다면 프로퍼티가 2개이상일 경우에는 어떻게 사용하는지 확인해보자.

```
<script language="JavaScript">
<!--
  var myJSONObject3 = {
    "test1": "hello1", "test2": "hello2", "test3":"hello3"
  }
//-->
</script>

<form>
  <input type="button" onclick="alert(myJSONObject3.test1)" value="click">
  <input type="button" onclick="alert(myJSONObject3.test2)" value="click">
  <input type="button" onclick="alert(myJSONObject3.test3)" value="click">
</form>
```

myJSONObject3의 각 프로퍼티인 "test1", "test2", "test3"를 각각 호출하면 각 프로퍼티의 값인 "hello1", "hello2", "hello3"를 꺼내오게된다.

■ 메소드

```
<script language="JavaScript">
<!--
  var myJSONObject4 = {
    "test1" : "function() { alert('This is method test1') }"
  }
//-->
</script>

<form>
  <input type="button" onclick="eval('a=' + myJSONObject4.test1); a()" value="click">
</form>
```

※ eval()함수는 변수를 javascript의 함수처럼 쓰는 명령어임.

myJSONObject4의 메소드 test1을 실행하는 예를 보여주고 있다. 메소드 test1을 실행하면 "This is method test1"을 확인할 수 있다.

■ 메소드(인수)

```
<script language="JavaScript">
<!--
  var myJSONObject5 = {
    "test2" : "function(arg) { alert('This is argument : ' + arg) } "
  }
//-->
</script>

<form>
  <input type="button" onclick="eval('var a=' + myJSONObject5.test2 + ''); a('hello');"
value="click">
```

```
  <input type="button" onclick="eval('(' + myJSONObject5.test2+')(₩'hello₩')');" value="click">
</form>
```

(1) eval('var a=' + myJSONObject5.test2 + ''); a('hello');
(2) eval('(' + myJSONObject5.test2+')(₩'hello₩')');

두가지 모두 동일한 결과를 보여주고 있으므로 사용하기 편한 형태를 골라 사용하면 된다.

[3] JSON의 데이터 타입(자료형)

JSON의 데이터 타입은 다음과 같다.

- string
- number
- boolean
- null
- array
- object

JSON Object의 각 프로퍼티 자료형은 "typeof"를 통해 확인이 가능하며, 각 자료형을 실제로 이용하는 예제를 통해 알아보기로 하자.

아래의 각 예제들은 typeof를 통해 오브젝트 또는 프로퍼티의 자료형을 확인하고, 오브젝트의 프로퍼티 값을 확인하는 예제이다.

■ string

string은 큰따옴표안에 둘러 싸인 zero 이상 Unicode 문자들의 조합이며, 쌍다옴표안에 감싸지며,backslash escape가 적용된다. 하나의 문자(character)도 하나의 문자열(character string)로서 표현된다. string은 C 또는 Java 문자열 처럼 매우 많이 비슷하다.

```
<script language="JavaScript">
<!--
  var obj1 = {
    "test" : "abc"
  };
//-->
</script>

<input type="button" onclick="alert(typeof obj1.test)" value="click">
<input type="button" onclick="alert(obj1.test)" value="click">
```

■ number

number는 8진수와 16진수 형식을 사용하지 않는것을 제외하면 C와 Java number 처럼 매우 많이 비슷하다.

```
<script language="JavaScript">
<!--
  var obj2 = {
    "test" : 123
  };
//-->
</script>

<input type="button" onclick="alert(typeof obj2.test)" value="click">
```

```
<input type="button" onclick="alert(obj2.test)" value="click">
```

■ boolean

boolean은 true/false의 값을 사용하며, C나 java의 boolean형과 비슷하다.

```
<script language="JavaScript">
<!--
  var obj3 = {
    "test" : true
  };
//-->
</script>

<input type="button" onclick="alert(typeof obj3.test)" value="click">
<input type="button" onclick="alert(obj3.test)" value="click">
```

■ null

null은 어떠한 형태를 담기 이전의 상태로, object로 취급받게 되며, 데이터가 할당되면 할당된 데이터의 타입에 따라 다시 구분되게 된다.

```
<script language="JavaScript">
<!--
  var obj4 = {
    "test" : null
  };
//-->
</script>

<input type="button" onclick="alert(typeof obj4.test)" value="click">
<input type="button" onclick="alert(obj4.test)" value="click">
<input type="button" onclick="obj4.test=false; alert(obj4.test)" value="click">
```

■ array

array은 값들의 순서화된 collection 이다. array는 [(left bracket)로 시작해서] (right bracket)로 끝내어 표현한다. , (comma)로 array의 값들을 구분한다.

1) 1차원 배열

```
<script language="JavaScript">
<!--
  var obj5 = [
    "test"
  ]
//-->
</script>

<input type="button" onclick="alert(typeof obj5[0])" value="click">
<input type="button" onclick="alert(obj5[0])" value="click">
```

2) 2차원 배열

```
<script language="JavaScript">
<!--
  var obj6 = {
    "test" : [
```

```
      "ccc", "ddd"
    ]
  }
//-->
</script>

<input type="button" onclick="alert(typeof obj6.test)" value="click">
<input type="button" onclick="alert(obj6.test[0])" value="click">
<input type="button" onclick="alert(obj6.test[1])" value="click">
```

■ object

object는 name/value 쌍들의 비순서화된 SET이다. object는 { (좌 중괄호)로 시작하고 } (우 중괄호)로 끝내어 표현한다. 각 name 뒤에 : (colon)을 붙이고 , (comma)로 name/value 쌍들 간을 구분한다.

```
<script language="JavaScript">
<!--
  var obj7 = {
    "test" : {
      "name" : "k2club",
      "id" : 123
    }
  }
//-->
</script>

<input type="button" onclick="alert(typeof obj7.test)" value="click">
<input type="button" onclick="alert(obj7.test.name)" value="click">
<input type="button" onclick="alert(obj6.test.id)" value="click">
```

※ object의 경우 array의 2차원 구조와 형태가 상당히 비슷하므로 주의하도록 하자.