JSON과 XML

JSON이란?

- JSON은 JavaScript Object Notation의 약자입니다.
- JSON은 사람이 읽을 수 있는 텍스트 기반의 데이터 교환 표준입니다.
- JSON은 XML의 대안으로서 좀 더 쉽게 데이터를 교환하고 저장하기 위하여 고안되었습니다.
- 또한, JSON은 텍스트 기반이므로 어떠한 프로그래밍 언어에서도 JSON 데이터를 읽고 사용할 수 있습니다.

JSON의 특징

1. JSON은 자바스크립트를 확장하여 만들어졌습니다.
2. JSON은 자바스크립트 객체 표기법을 따릅니다.
3. JSON은 사람과 기계가 모두 읽기 편하도록 고안되었습니다.
4. JSON은 프로그래밍 언어와 운영체제에 독립적입니다.

XML이란?

- XML은 Extensible Markup Language의 약자입니다.
- XML은 HTML과 매우 비슷한 문자 기반의 마크업 언어(text-based markup language)입니다.
- 이 언어는 사람과 기계가 동시에 읽기 편한 구조로 되어 있습니다.
- XML은 HTML처럼 데이터를 보여주는 목적이 아닌, 데이터를 저장하고 전달할 목적으로만 만들어졌습니다.
- 또한, XML 태그는 HTML 태그처럼 미리 정의되어 있지 않고, 사용자가 직접 정의할 수 있습니다.

JSON과 XML의 공통점

1. 둘 다 데이터를 저장하고 전달하기 위해 고안되었습니다.
2. 둘 다 기계뿐만 아니라 사람도 쉽게 읽을 수 있습니다.
3. 둘 다 계층적인 데이터 구조를 가집니다.
4. 둘 다 다양한 프로그래밍 언어에 의해 파싱될 수 있습니다.
5. 둘 다 XMLHttpRequest 객체를 이용하여 서버로부터 데이터를 전송받을 수 있습니다

JSON과 XML의 차이점

1. JSON은 종료 태그를 사용하지 않습니다.
2. JSON의 구문이 XML의 구문보다 더 짧습니다.
3. JSON 데이터가 XML 데이터보다 더 빨리 읽고 쓸 수 있습니다.
4. XML은 배열을 사용할 수 없지만, JSON은 배열을 사용할 수 있습니다.
5. XML은 XML 파서로 파싱되며, JSON은 자바스크립트 표준 함수인 eval() 함수로 파싱됩니다

XML 예시

```
<dog>
    <name>식빵</name>
    <family>웰시코기<family>
    <age>1</age>
    <weight>2.14</weight>
</dog>
```

JSON 예시

```
//JSON 객체
{
    "name": "식빵",
    "family": "웰시코기",
    "age": 1,
    "weight": 2.14
}
```

```
//JSON 배열
"dog": [
    {
        "name": "식빵",
        "family": "웰시코기",
        "age": 1, "weight": 2.14
    },
    {
        "name": "콩콩",
        "family": "포메라니안",
        "age": 3, "weight": 2.5
      },
    {
        "name": "젤리",
        "family": "푸들",
        "age": 7, "weight": 3.1
       }
]
```

JSON 스키마

- JSON은 좀 더 쉽게 데이터를 교환하고 저장하기 위하여 만들어진 데이터 교환 표준입니다.
- 이때 JSON 데이터를 전송받는 측에서는 전송받은 데이터가 적법한 형식의 데이터인지를 확인할 방법이 필요합니다.
- 따라서 적법한 JSON 데이터의 형식을 기술한 문서를 JSON 스키마(schema)라고 합니다.

JSON 스키마는 다음과 같은 세 가지 검증 과정을 거칩니다.

1. 데이터의 타입이 정확한가?

2. 필수로 받아와야 하는 데이터가 포함되어 있는가?

3. 데이터가 원하는 범위 안에 있는가?

검증 키워드	설명
type	유효한 데이터의 타입을 명시함.
properties	유효한 데이터 이름과 값의 쌍들을 명시함.
required	명시한 배열의 모든 요소를 프로퍼티로 가지고 있어야만 유효함.
minimum	최소값 이상의 숫자만 유효함.
maximum	최댓값 이하의 숫자만 유효함.
multipleOf	명시한 숫자의 배수만 유효함.
maxLength	명시한 최대 길이 이하의 문자열만 유효함.
minLength	명시한 최소 길이 이상의 문자열만 유효함.
pattern	명시한 정규 표현식에 해당하는 문자열만 유효함.

스키마에 대한 정보를 나타내는 메타 데이터(metadata) 키워드는 다음과 같습니다.

1. title
2. description
3. default

```
{
    "title": "강아지 스키마",
    "description": "이 스키마는 강아지에 관한 데이터를 검증하기 위해
작성된 스키마임.",
    "type": "object",
    "properties": {
        "name": {"type": "string"},
        "family": {"type": "string"},
        "age": {"type": "integer"},
        "weight": {"type": "number"},
        "owner": {
            "type": "object",
            "properties": {
                "ownerName": {"type": "string"},
                "phone": {"type": "string"}
            }
        }
```

```
    }
}
```

JSON.stringify() 메소드

JSON.stringify() 메소드는 인수로 전달받은 자바스크립트 객체를 문자열로 변환하여 반환합니다.

문법

```
JSON.stringify(value)
```

예제

```
<p id="stringify"></p>

<script>
// 자바스크립트 객체
let dog = {name: "식빵", family: "웰시코기", age: 1, weight:
2.14};

// 자바스크립트 객체를 문자열로 변환함.
let data = JSON.stringify(dog);

document.getElementById("stringify").innerHTML = data;
</script>
```

JSON.parse() 메소드

JSON.parse() 메소드는 JSON.stringify() 메소드와는 반대로 인수로 전달받은 문자열을 자바스크립트 객체로 변환하여 반환합니다.

문법

```
JSON.parse(text) //text에는 JSON형식의 문자열이 전달 되야 합니다
```

예시

```
<p id="parse"></p>

<script>
// JSON 형식의 문자열
let data = '{"name": "식빵", "family": "웰시코기", "age": 1,
"weight": 2.14}';

// JSON 형식의 문자열을 자바스크립트 객체로 변환함.
let dog = JSON.parse(data);

document.getElementById("parse").innerHTML = dog + "<br>";
document.getElementById("parse").innerHTML += dog.name + ",
" + dog.family;
</script>
```

---

toJSON() 메소드

자바스크립트의 toJSON() 메소드는 자바스크립트의 Date 객체의 데이터를 JSON 형식의 문자열로 변환하여 반환합니다.

따라서 이 메소드는 Date.prototype 객체에서만 사용할 수 있습니다.

toJSON() 메소드는 접미사 Z로 식별되는 UTC 표준 시간대의 날짜를 ISO 8601 형식의 문자열로 반환합니다.

예시

```
<p id="toJson"></p>

<script>
// 자바스크립트 Date 객체
let date = new Date();
```

```
// Date 객체를 JSON 형식의 문자열로 변환함.
let str = date.toJSON();

document.getElementById("json").innerHTML = date + "<br>";
document.getElementById("json").innerHTML += str;
</script>
```