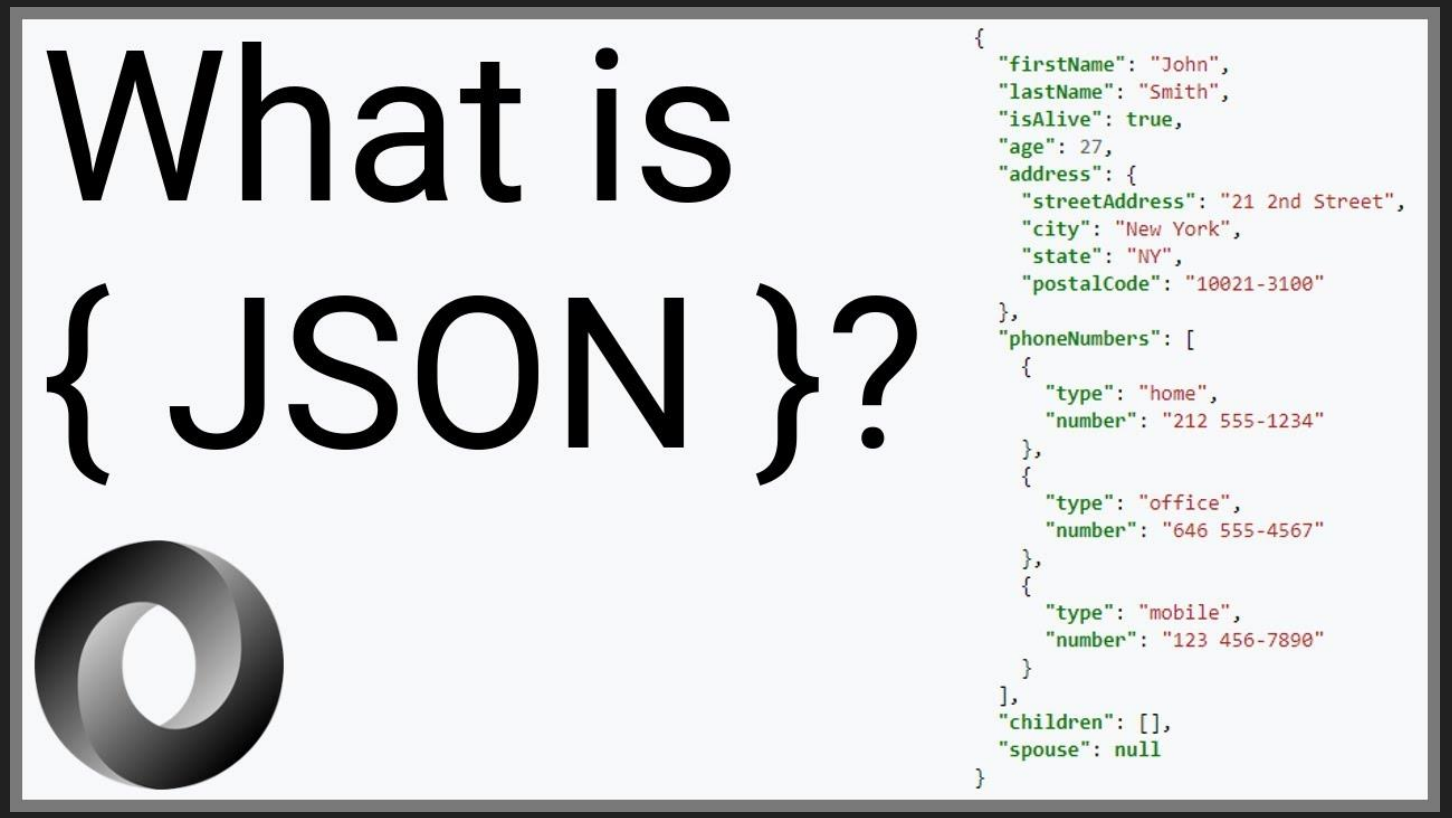
JavaScript

JSON – 김근형 강사

JSON

- JSON(**J**ava**S**cript **O**bject **N**otation)



JSON

- JSON(**J**ava**S**cript **O**bject **N**otation)
  - JSON은 데이터 저장 및 전송을 위한 텍스트 형식.
  - JSON은 자바스크립트를 확장하여 만들어 짐.
  - JSON은 자바스크립트 객체 표기법을 따른다.
  - JSON은 사람과 기계가 모두 읽기 편하도록 고안되었음.
  - JavaScript 객체 표기법으로 작성된 일반 텍스트이다.
  - JSON은 컴퓨터 간에 데이터를 전송하는 데 사용된다.
  - JSON은 **언어 독립적**이다
    - JSON 구문은 JavaScript 객체 표기법에서 파생되었지만 데이터 텍스트만을 가지고 있다.
    - 많은 프로그래밍 언어가 JSON을 읽고 쓰는 방법을 지원한다.

JSON Syntax

- JSON(**J**ava**S**cript **O**bject **N**otation) 작성 방법
  - 데이터는 이름 / 값의 쌍으로 존재한다.
  - JSON 데이터는 데이터 이름, 콜론(:), 값의 순서로 구성한다.
  - 데이터의 쌍은 쉼표로 구분한다.
  - 중괄호는 객체를 표현한다.
  - 대괄호는 배열을 나타낸다.
  - 키도 문자열이므로 “”/’’ 로 표현한다.

JSON Type

- JSON(**J**ava**S**cript **O**bject **N**otation) 에서 사용할 수 있는 자바 스크립트 타입
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - null

- JSON(**J**ava**S**cript **O**bject **N**otation) 에서 사용할 수 없는 자바 스크립트 타입
  - a function
  - a date
  - undefined

JSON Type

- JSON 타입 예제

```
{
    "String" : {"name":"John"},
    "Number" : {"age":30},
    "Object" : {"name":"John", "age":30, "city":"New York"},
    "Array" : ["John", "Anna", "Peter"],
    "Boolean" : {"sale":true},
    "Null" : {"middlename":null}
}
```

JSON Parse

- JSON.parse()
  - JSON.parse() 메서드는 JSON 문자열의 구문을 분석하고, 그 결과에서 JavaScript 값이나 객체를 생성한다.
  - 선택적으로, reviver 함수를 인수로 전달할 경우, 결과를 반환하기 전에 변형할 수 있다.

```
JSON.parse(text[, reviver])
```

  - text : JSON으로 변환할 문자열
  - reviver : 함수라면, 변환 결과를 반환하기 전에 이 인수에 전달해 변형함.

JSON Parse

- JSON.parse() – ex1

```
let a = JSON.parse('{}');                // {}
console.log(a);
let b = JSON.parse('true');              // true
console.log(b);
let c = JSON.parse('"foo"');             // "foo"
console.log(c);
let d = JSON.parse('[1, 5, "false"]'); // [1, 5, "false"]
console.log(d);
let e = JSON.parse('null');              // null
console.log(e);
```

JSON Parse

- JSON.parse() – JSON으로 변형한 객체에 접근하는 방법

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

<script>
    const txt = '{"name":"John", "age":30, "city":"New York"}'
    const obj = JSON.parse(txt);
    document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>
```

John, 30

JSON Parse

- JSON.parse() – 문자열로 온 시간을 사용하는 방법

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

<script>
    const text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}';
    const obj = JSON.parse(text);
    obj.birth = new Date(obj.birth);
    document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth;
</script>
```

John, Sun Dec 14 1986 09:00:00 GMT+0900 (대한민국 표준시)

JSON Parse

- JSON.parse() – 문자열로 온 시간을 사용하는 방법(reviver 이용)

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

<script>
    const text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}';
    const obj = JSON.parse(text, function (key, value) {
        if (key == "birth") {
            return new Date(value);
        } else {
            return value;
        }
    });
    document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth;
</script>
```

John, Sun Dec 14 1986 09:00:00 GMT+0900 (대한민국 표준시)

JSON stringify

- JSON.stringify()
  - JSON.stringify() 메서드는 JavaScript 값이나 객체를 JSON 문자열로 변환한다.
  - 선택적으로, replacer를 함수로 전달할 경우 변환 전 값을 변형할 수 있고, 배열로 전달할 경우 지정한 속성만 결과에 포함한다.

```
JSON.stringify(value[, replacer[, space]])
```

JSON stringify

- JSON.stringify()
  - value : JSON 문자열로 변환할 값
  - replacer : 문자열화 동작 방식을 변경하는 함수, 혹은 JSON 문자열에 포함될 값 객체의 속성들을 선택하기 위한 화이트리스트(whitelist)로 쓰이는 String 과 Number 객체들의 배열. 이 값이 null 이거나 제공되지 않으면, 객체의 모든 속성들이 JSON 문자열 결과에 포함된다.
  - space : 가독성을 목적으로 JSON 문자열 출력에 공백을 삽입하는데 사용되는 String 또는 Number 객체. 이것이 Number 라면, 공백으로 사용되는 스페이스(space)의 수를 나타낸다; 이 수가 10 보다 크면 10 으로 제한된다. 1 보다 작은 값은 스페이스가 사용되지 않는 것을 나타낸다. 이것이 String 이라면, 그 문자열(만약 길이가 10 보다 길다면, 첫번째 10 개의 문자)이 공백으로 사용된다. 이 매개 변수가 제공되지 않는다면(또는 null 이면), 공백이 사용되지 않는다.

JSON stringify

- JSON.stringify() – 예제1

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

<script>
    const obj = { name: "John", age: 30, city: "New York" };
    const myJSON = JSON.stringify(obj);
    document.getElementById("demo").innerHTML = myJSON;
</script>
```

{"name":"John","age":30,"city":"New York"}

JSON stringify

- JSON.stringify() – 예제2

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

<script>
    const arr = ["John", "Peter", "Sally", "Jane"];
    const myJSON = JSON.stringify(arr);
    document.getElementById("demo").innerHTML = myJSON;
</script>
```

["John","Peter","Sally","Jane"]

JSON stringify

- JSON.stringify() – localStorage로 저장 후에 꺼내와서 사용하는 방법

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

<script>
    // Storing data:
    const myObj = { name: "John", age: 31, city: "New York" };
    const myJSON = JSON.stringify(myObj);
    localStorage.setItem("testJSON", myJSON);

    // Retrieving data:
    let text = localStorage.getItem("testJSON");
    let obj = JSON.parse(text);
    document.getElementById("demo").innerHTML = obj.name;
</script>
```

John

JSON stringify

- JSON.stringify() – 현재 시간을 문자열 화 시키는 방법

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

<script>
    const obj = { name: "John", today: new Date(), city: "New York" };
    const myJSON = JSON.stringify(obj);
    document.getElementById("demo").innerHTML = myJSON;
</script>
```

{"name":"John","today":"2021-07-03T15:27:12.250Z","city":"New York"}

JSON stringify

- JSON.stringify() – replacer 함수 예제

```
function replacer(key, value) {
    if (typeof value === "string") {
        return undefined;
    }
    return value;
}

var foo = { foundation: "Mozilla", model: "box", week: 45, transport: "car", month: 7 };
var jsonString = JSON.stringify(foo, replacer);
console.log(jsonString);
```

```
{"week":45,"month":7}
```

JSON stringify

- JSON.stringify() – replacer 배열 예제

```
<script>
    var foo = {foundation: "Mozilla", model: "box", week: 45, transport: "car", month: 7};
    var jsonString = JSON.stringify(foo, ['week', 'month']);
    console.log(jsonString);
</script>
```

```
{"week":45,"month":7}
```

JSON stringify

- JSON.stringify() – space 예제

```
var x1 = JSON.stringify({ a: 2 }, null, ' ');
console.log(x1);
// '{
//   "a": 2
// }'
var x2 = JSON.stringify({ uno: 1, dos: 2 }, null, '\t');
console.log(x2);
// returns the string:
// '{
//     "uno": 1,
//     "dos": 2
// }'
```