

2023.04.25

CORE JAVASCRIPT

JSON과 메서드

stringify

parse

JSON

데이터를 저장하거나 전송할 때 많이 사용하는 경량의 데이터 교환 형식

JSON.stringify

객체를 JSON으로 변환

JSON.parse

JSON을 객체로 변환

stringify

객체를 JSON으로 변환

```
1 let student = {
2     name: "John",
3     age: 30,
4     isAdmin: false,
5     courses: ["html", "css", "js"],
6     wife: null,
7 };
8 let json = JSON.stringify(student);
9
10 console.log(json);
11 /*
12 {
13   "name": "John",
14   "age": 30,
15   "isAdmin": false,
16   "courses": ["html", "css", "js"],
17   "wife": null
18 }
19 */
20 console.log(typeof json); // string
```


stringify

객체를 JSON으로 변환

```
1  let student = {
2      name: "John",
3      age: 30,
4      isAdmin: false,
5      courses: ["html", "css", "js"],
6      wife: null,
7  };
8  let json = JSON.stringify(student);
9
10 console.log(json);
11 /*
12 {
13   "name": "John",
14   "age": 30,
15   "isAdmin": false,
16   "courses": ["html", "css", "js"],
17   "wife": null
18 }
19 */
20 console.log(typeof json); // string
```

type - 문자열

stringify



```
1 // 숫자를 JSON으로 인코딩하면 숫자입니다.
2 alert(JSON.stringify(1)); // 1
3
4 // 문자열을 JSON으로 인코딩하면 문자열입니다(다만, 큰따옴표가 추가됩니다).
5 alert(JSON.stringify("test")); // "test"
```

적용가능 자료형

1. 객체 { ... }
2. 배열 [...]
3. 원시형:
문자형 / 숫자형 / true와 false / null

stringify

```
1 let user = {
2     sayHi() {
3         // 무시
4         alert("Hello");
5     },
6     [Symbol("id")]: 123, // 무시
7     something: undefined, // 무시
8 };
9
10 alert(JSON.stringify(user)); // {} (빈 객체가 출력됨)
```

적용불가능 자료형

1. 함수 프로퍼티 (메서드)
2. 심볼형 프로퍼티 (키가 심볼인 프로퍼티)
3. 값이 **undefined**인 프로퍼티

stringify**장점****1 중첩 객체도 알아서 문자열로 바꿔준다**

```
1 let meetup = {  
2   title: "Conference",  
3   room: {  
4     number: 23,  
5     participants: ["john", "ann"],  
6   },  
7   date: new Date(Date.UTC(2017, 0, 1)),  
8 };
```

```
1 console.log(JSON.stringify(meetup));  
2 /*  
3  {  
4    "title": "Conference",  
5    "room": { "number": 23, "participants": ["john", "ann"] },  
6    "date": "2017-01-01T00:00:00.000Z",  
7  }  
*/
```


stringify**장점**

```
1 let meetup = {  
2   title: "Conference",  
3   room: {  
4     number: 23,  
5     participants: ["john", "ann"],  
6   },  
7   date: new Date(Date.UTC(2017, 0, 1)),  
8 };
```

- 1 중첩 객체도 알아서 문자열로 바꿔준다
- 2 Date 객체는 자동으로 문자열로 변환된다
(내장 메서드 toJSON이 호출된다)



```
1 console.log(JSON.stringify(meetup));  
2 /*  
3 {  
4   "title": "Conference",  
5   "room": {"number": 23, "participants": ["john", "ann"]},  
6   "date": "2017-01-01T00:00:00.000Z",  
7 }*/
```

stringify

2,3번째 파라미터



```
1 let json = JSON.stringify(value[, replacer, space])
```

- **value** - 인코딩 하려는 값
- **replacer** - 인코딩 하길 원하는 프로퍼티가 담긴 **배열 또는 매핑 함수**
- **space** - 서식 변경 목적으로 사용할 공백 문자 수

stringify**2번째 파라미터 - 배열**

replacer - 인코딩 하길 원하는 프로퍼티가 담긴 **배열** 또는 **매핑 함수**

```
1 let room = {
2     number: 23,
3 };
4
5 let meetup = {
6     title: "Conference",
7     participants: [{ name: "John" }, { name: "Alice" }],
8     place: room, // meetup은 room을 참조합니다.
9 };
10
11 room.occupiedBy = meetup; // room references meetup
12
13 alert(JSON.stringify(meetup, ["title", "participants", "name"])); //원하는 것만 두번째 인자로 배열형식으로
14 // {"title":"Conference","participants":[{"name":"John"},{"name":"Alice"}]}
```

stringify**2번째 파라미터 - 배열**

replacer - 인코딩 하길 원하는 프로퍼티가 담긴 배열 또는 매핑 함수

```
1 let room = {
2   number: 23,
3 };
4
5 let meetup = {
6   title: "Conference",
7   participants: [{ name: "John" }, { name: "Alice" }],
8   place: room, // meetup은 room을 참조합니다.
9 };
10
11 room.occupiedBy = meetup; // room references meetup
12
13 alert(JSON.stringify(meetup, ["title", "participants", "name"])); //원하는 것만 두번째 인자로 배열형식으로
14 // {"title":"Conference","participants":[{"name":"John"}, {"name":"Alice"}]}
```

중첩된 프로퍼티 이름도 넣어야 출력된다!

stringify**2번째 파라미터 - 배열**

```
1 let room = {
2     number: 23,
3 };
4
5 let meetup = {
6     title: "Conference",
7     participants: [{ name: "John" }, { name: "Alice" }],
8     place: room, // meetup은 room을 참조합니다.
9 };
10
11 room.occupiedBy = meetup; // room references meetup
12
13 alert(JSON.stringify(meetup, ["title", "participants", "name", "place", "number"]));
14 /*
15 {
16     "title":"Conference",
17     "participants":[{"name":"John"},{"name":"Alice"}],
18     "place":{"number":23}
19 }
20 */
```

순환참조를 발생시키는 프로퍼티 -> stringify X

stringify**2번째 파라미터 - 배열**

```
1 let room = {
2   number: 23,
3 };
4
5 let meetup = {
6   title: "Conference",
7   participants: [{ name: "John" }, { name: "Alice" }],
8   place: room, // meetup은 room을 참조합니다.
9 };
10
11 room.occupiedBy = meetup; // room references meetup
12
13 alert(JSON.stringify(meetup, ["title", "participants", "name", "place", "number"]));
14 /*
15 {
16   "title": "Conference",
17   "participants": [{"name": "John"}, {"name": "Alice"}],
18   "place": {"number": 23}
19 }
20 */
```

순환참조를 발생시키는 프로퍼티 -> stringify X
room.occupiedBy만 제외한 모든 프로퍼티를 배열에 넣으면 출력된다

stringify**2번째 파라미터 - 배열**

```
1  let room = {
2      number: 23,
3  };
4
5  let meetup = {
6      title: "Conference",
7      participants: [{ name: "John" }, { name: "Alice" }],
8      place: room, // meetup은 room을 참조합니다.
9  };
10
11 room.occupiedBy = meetup; // room references meetup
12
13 alert(JSON.stringify(meetup, ["title", "participants", "name", "place", "number"]));
14 /*
15 {
16     "title":"Conference",
17     "participants":[{"name":"John"},{"name":"Alice"}],
18     "place":{"number":23}
19 }
20 */
```

순환참조를 발생시키는 프로퍼티 -> stringify X
room.occupiedBy만 제외한 모든 프로퍼티를 배열에 넣으면 출력된다

stringify**2번째 파라미터 - 함수**

```
1 alert(  
2     JSON.stringify(meetup, function replacer(key, value) {  
3         alert(`${key}: ${value}`);  
4         return key == "occupiedBy" ? undefined : value;  
5     })  
6 );
```

replacer 함수
function (key, value)

| | |
|-------------|---------------------------------|
| : | [object Object] |
| title: | Conference |
| occupiedBy: | [object Object],[object Object] |
| 0: | [object Object] |
| name: | John |
| 1: | [object Object] |
| name: | Alice |
| place: | [object Object] |
| number: | 23 |
| occupiedBy: | [object Object] |
| self: | [object Object] |

stringify**2번째 파라미터 - 함수**

```
1 alert(  
2     JSON.stringify(meetup, function replacer(key, value) {  
3         alert(`${key}: ${value}`);  
4         return key == "occupiedBy" ? undefined : value;  
5     })  
6 );
```

replacer 함수
function (key, value)

특정 프로퍼티를 누락시키려면 반환 값을 **undefined**로 만들면 된다!

stringify

3번째 파라미터 - 숫자

space - 공백 문자 수

```
1 let user = {  
2   name: "John",  
3   age: 25,  
4   roles: {  
5     isAdmin: false,  
6     isEditor: true,  
7   },  
8 };  
9  
10 alert(JSON.stringify(user));  
11 alert(JSON.stringify(user, null, 2)); //줄바꿈, 들여쓰기됨
```

이 페이지 내용:

```
{ "name": "John", "age": 25, "roles": { "isAdmin": false, "isEditor": true } }
```

확인

이 페이지 내용:

```
{  
  "name": "John",  
  "age": 25,  
  "roles": {  
    "isAdmin": false,  
    "isEditor": true  
  }  
}
```

확인

parse

JSON을 객체로 변환



```
1 let userData = '{ "name": "John", "age": 35, "isAdmin": false, "friends": [0,1,2,3] }';  
2  
3 let user = JSON.parse(userData);  
4  
5 alert(user.friends[1]); // 1
```

parse

주의사항

```
1 let json = `{
2     name: "John",           // 실수 1: 프로퍼티 이름을 큰따옴표로 감싸지 않았습니다.
3     "surname": 'Smith',     // 실수 2: 프로퍼티 값은 큰따옴표로 감싸야 하는데, 작은따옴표로 감쌌습니다.
4     'isAdmin': false       // 실수 3: 프로퍼티 키는 큰따옴표로 감싸야 하는데, 작은따옴표로 감쌌습니다.
5     "birthday": new Date(2000, 2, 3), // 실수 4: "new"를 사용할 수 없습니다. 순수한 값(bare value)만 사용할 수 있습니다.
6     "friends": [0,1,2,3]    // 이 프로퍼티는 괜찮습니다.
7 }`;
```

- 프로퍼티의 키, 값 모두 문자열일 경우 “큰따옴표”
- 순수한 값만 사용 가능 (new 객체 안됨)
- JSON은 위와 같은 주석 작성 X

parse**2번째 파라미터**

reviver - 변환 결과를 반환하기 전에 이 인수에 전달해 변형하는 함수

```
1 let str = '{"title":"Conference","date":"2017-11-30T12:00:00.000Z"}';
2
3 let meetup = JSON.parse(str);
4
5 alert(meetup.date.getDate()); // 에러!
6
7 // 해결책
8 let meetup = JSON.parse(str, function (key, value) {
9     if (key == "date") return new Date(value);
10    return value;
11 });
12
13 alert(meetup.date.getDate()); // 이제 제대로 동작!
```

단순 문자열로 인식



Date 메서드 사용 불가

parse**2번째 파라미터**

reviver - 변환 결과를 반환하기 전에 이 인수에 전달해 변형하는 함수

```
1 let str = '{"title":"Conference","date":"2017-11-30T12:00:00.000Z"}';
2
3 let meetup = JSON.parse(str);
4
5 alert(meetup.date.getDate()); // 에러!
6
7 // 해결책
8 let meetup = JSON.parse(str, function (key, value) {
9     if (key == "date") return new Date(value);
10    return value;
11 });
12
13 alert(meetup.date.getDate()); // 이제 제대로 동작!
```

단순 문자열로 인식



Date 메서드 사용 불가



reviver 함수로 조건에 따라

Date 객체로 변환



Date 메서드 사용 가능

THANK YOU